OZIX Technical Summary

Version: 0.5

30 October 1989

Issued by: DECwest Engineering

digital™

Acknowledgements

This document represents the original work of many individuals on the OZIX project. Key contributors include the following.

Document Planning and Production

Marilyn Fries, John Penney, Gordon Furbush, Bill Talcott

Chapter Authors

Introduction	Gordon Furbush Jeff Havens
User Environment and Application Environment	Myles Connors
System Administration	Mark Ditto
Resource/Fault Management	David Walp
Security	Jim Schirmer
Internationalization	Claire Cockcroft
Base System Architecture	Chris Saether
I/O System	Sumanta Chatterjee Debbie Girdler
Terminal Subsystems	Sumanta Chatterjee
Network Implementation Architecture	OZIX Networks Group
System Installation and System Directory Layout	Mark Ditto
Performance Engineering	Pete Benoit Mike Peterson
Software Quality and Testing Strategy	Jan D'Addamio

TABLE OF CONTENTS

FIGURES

TABLES

Preface

This document provides a technical summary of the OZIX system. For a marketing analysis of the OZIX program, see the *OZIX Vision Document*.

CHAPTER 1

INTRODUCTION

1.1 The OZIX Project

OZIX is an extensible, hardware-independent operating system that integrates modern technologies for distributed systems, fault tolerance, data integrity, and high-performance I/O. OZIX is structured to fully support the process management, file processing, system calls, and communication services defined in the POSIX™, X/OPEN™, OSF™, and ISO standards. The reliability, data integrity, and security requirements of transaction processing applications are also supported as core features in the OZIX system.

The entire complement of OZIX features will be implemented over a series of releases. Such features include system elements, such as database and transaction processing software, compilers, tools, and utilities from other Digital groups, in addition to selected elements from third-party vendors.

Version 1 of OZIX is a high-performance production system targeted at the open systems market. While transaction processing technologies are built into Version 1, these capabilities are not made visible to higher levels of software until later versions. Version 1 is designed to function as a network and compute server in a multivendor, distributed systems network. Version 1 serves this network by providing file services (via NFS™) and compute services (RPC and user applications). Version 1 also includes integrated system and network management features, license management, and CDROM distribution. Portability of applications is provided with application integration architecture (AIA) components such as DECwindows™ client, Compound Document Architecture (CDA), and Concert Multithread Architecture (CMA).

Project Priorities

In order to maintain a productive focus, the goals of the OZIX project are prioritized in the following order:

1. OSF/AES and ULTRIX™ interoperability
2. Seven-days-a-week, Twenty-four-hours-a-day (7x24) Reliability
3. Performance, Robustness
4. B2 Security
5. Internationalization
6. Transaction Processing Primitives
7. IBM® Interoperability
8. Cost
9. Time to Market
10. UNIX® Interoperability

11. Portability
12. VMS™ Interoperability

1.2 Building the Foundation of a Modern Operating System

Hardware and system designs have progressed significantly over the last decade. The fundamental assumptions made when designing operating systems 10 years ago – 1 VUP CPUs and 256 kilobytes of memory—no longer apply to today's systems. These assumptions have made it very difficult to retrofit new concepts into existing operating systems.

Back in 1978, higher performance processors were 1 VUP, 2 megabytes of memory was considered more than enough, demands on disk storage were low, networks were small, local area networks didn't even exist, and no one was concerned about security or internationalization. Today, we have hardware offering 50+ VUPs that can support hundreds of megabytes of memory; users are pushing the limits of disk storage; networks must support thousands of nodes consisting of systems from multiple vendors; there is an urgent demand for security; and the fastest growth in computer sales is taking place in the international markets. Further, customers are requiring standard user interfaces on systems that deliver the highest performance possible.

The OZIX project offers unique opportunities to deliver a system that can keep pace with these rapid improvements in technology and increasing customer needs. Many of these opportunities stem from building a contemporary operating system that is based on modern concepts, along with an understanding of the current and future demands of the production system marketplace. The OZIX system, however, is as much a product of design methodologies as it is of modern technologies. As an operating system built from the ground up, every component is methodically designed and tested to ensure that performance, reliability, security, system management, and internationalization are cleanly integrated into the overall structure of the system.

1.3 Building an Open System

OZIX is a modern operating system platform that integrates the existing and emerging open system standards defined by POSIX, X/OPEN, OSF, and ISO/OSI. Support for standards—such as those defining user interface, data exchange, networking, file access, multi-processing, transaction processing, security, and internationalization—are pervasive throughout the OZIX system. Particular efforts are made to support the applications, networking, and system/network management standards necessary to ensure the success of OZIX in distributed system environments. Conformance testing ensures that the OZIX implementation of these standards conforms to the standards definitions.

1.4 The OZIX Architecture: A Structured Approach

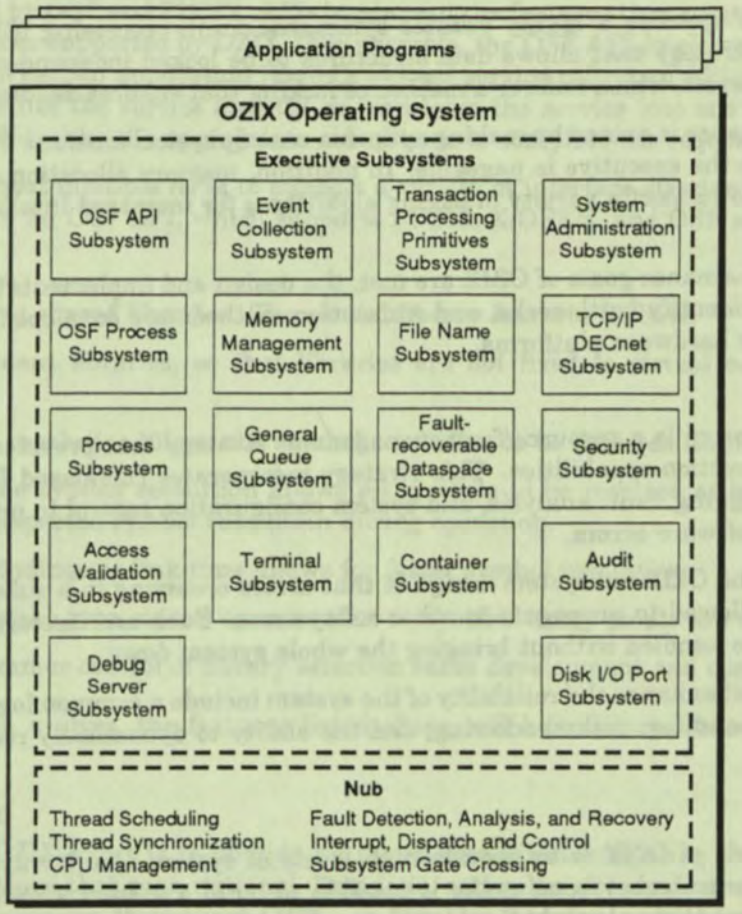
The OZIX architecture defines a highly structured environment within which system components are easily designed and implemented to deliver maximum performance, reliability, flexibility, and security. The primary goal of the OZIX system is to provide a solid foundation for transaction processing (TP) and customized applications. To accomplish this goal, particular emphasis is placed on supporting modern fault recovery and I/O technologies that provide superior availability and integrity of data across a large number of storage devices.

In order to ensure that OZIX remains competitive through the 1990s and beyond, the OZIX system is modular and extensible enough to adapt to new technologies and interface standards as they evolve.

Subsystems

As illustrated in Figure 1, the OZIX architecture defines functional units called *subsystems*. Each subsystem is implemented as part of the executive to support a specific functional area in the OS. Separate subsystems are used, for example, to support application programming interfaces, the file system, memory management, I/O devices, and so on.

Figure 1: OZIX Operating System



Subsystems reduce the complexity of the entire system by breaking it down into functional units. Each subsystem can be understood on an individual basis, without the need to understand all of the other subsystems in the operating system. Such functional division simplifies the task of design, as well as the task of updating the system to accommodate new features.

The Nub

The OZIX system is designed to keep hardware-dependent code to a minimum. Hardware-dependent operations—such as multiprocessor coordination, thread dispatching, condition handling, timer support, and so on—are implemented by a small, low-level component called the *nub*. By restricting all of the processor-dependent code to the nub, it is not necessary to modify the rest of the system when porting OZIX to new hardware.

OZIX Technical Summary

High Performance

OZIX is designed to make maximum use of multiple-processor, high-performance hardware. The OZIX architecture defines a very efficient multi-threading environment in which threads are created and destroyed quickly and with minimal overhead. OZIX supports a wide range of thread priority levels, including real-time priorities. In order to deliver fast real-time response to high priority requests, threads executing in the executive can be preempted by higher-priority threads. This OZIX multi-threading environment is complemented by OZIX compilers and runtime environment, which feature thread-safe libraries, minimized locking contention, and multi-thread debug capabilities.

The OZIX architecture specifies a highly reliable symmetric multi-processing (SMP) environment by defining a locking strategy that allows data structures to be locked independently. This locking strategy includes lock levels, which enforce a method of locking that ensures deadlock-free operation.

Overall system performance is gained by making optimum use of physical memory. For example, some of the code and data in the executive is pageable. In addition, memory allocation and management routines are provided to support a variety of paging algorithms for increased intelligence in memory paging.

To ensure that the performance goals of OZIX are met, the design and implementation of the system is closely monitored to identify bottlenecks, and simulation methods are developed for testing OZIX performance on various hardware platforms.

Reliable Operation

Integral to the OZIX design is a resource/fault management strategy that provides a wide range of error detection and correction capabilities. This strategy incorporates rule-based failure prediction, error detection, error logging, fault analysis, and system configuration control to predict, detect, and isolate hardware and software errors.

One of the benefits of the OZIX subsystem design is that errors occurring in a subsystem are easily isolated, and are not allowed to propagate to other subsystems. Such architectural firewalls allow subsystem failures to be handled without bringing the whole system down.

Other OZIX features that enhance the reliability of the system include a common logging mechanism, stack-based exception handling, disk shadowing, and the ability to dynamically reconfigure storage devices.

B2 Security

One of the primary goals of OZIX is to provide high levels of system security and integrity. The functional isolation between subsystems make it possible to build a system with levels of security and integrity enforcement that are unobtainable by any other means.

OZIX is designed to obtain a B2 level of certification by the Department of Defense (DoD). The constraints imposed by DoD security policies, however, are not suitable for most commercial applications. OZIX, therefore, supports multiple security and integrity policies to satisfy the needs of both government and commercial environments. Each of these security and integrity policies are enforced with the full strengths of a B2 system.

Maximum security and integrity enforcement has an impact on system performance. Since not all customers in all environments require the full B2-level of enforcement, the system can be tuned using a wide variety of enforcement parameters to provide the appropriate mix of performance, security, and integrity for a given customer environment.

Security is discussed in more detail in the Security chapter.

1.5 User and Application Support

OZIX supports standard user interfaces such as X/Windows and POSIX-compliant shells, commands, and utilities. In addition to these standards, the OZIX executive also incorporates Digital enhancements that integrate internationalization into the system.

Application Programming Interface (API) Subsystems

Application programming interface (API) subsystems serve a key role in implementing interfaces, such as those defined by OSF and POSIX. APIs implement the features that are unique to the various programming interfaces supported by OZIX. (For example, the OSF API implements file descriptors, signals, and so on.) When an application issues a system service call to an API subsystem, the API subsystem either handles the service directly, or translates the service into one or more subsystem procedure calls, which invoke the appropriate subsystems to complete the request.

OZIX is designed to host multiple APIs to support a variety of interface standards. OZIX, Version 1 is to be delivered with an OSF API, which supports POSIX, X/OPEN, and OSF applications.

Shared Libraries

The executive supports shared libraries. The shared library features include:

- Position-independent libraries, so that libraries are not fixed to virtual addresses at library compile time.
- Dynamic symbol resolution allows the symbol resolution to be deferred until actually used.
- Optional load-time symbol resolution allows all symbols to be resolved at image start-up time and prevents unexpected symbol resolution during operation.
- Partial default bindings at link time allows for faster symbol resolution.
- Version and interface type control ensures that compatible entry points are used.
- User and programmer control of library selection eases development and customization.

As the OSF definition evolves, the features listed above will be enhanced to conform to the OSF interface definition.

International Support

A major goal of the OZIX effort is to provide an internationalized computing environment capable of supporting applications that span national, linguistic, and cultural boundaries. Such internationalization support is pervasive throughout OZIX—in libraries, terminal services, the file system, the base system architecture, and the message facility. OZIX supports the characters for most every language through the implementation of the multiple octet character set (MOCS). In MOCS, up to 4 bytes can be used to define each character to support large character sets, such as Japanese Kanji, Chinese Hanzi, and South Korean Hangul and Hanja.

OZIX allows multiple libraries to be built on top of the OSF API subsystem. A standard C library is provided to support existing applications based on 8-bit character sets, as well as libraries utilizing compound strings to support new international applications for world-wide markets.

Internationalization is discussed in more detail in the Internationalization chapter.

1.6 Transaction Processing Primitives

The executive defines services that provide low-level transaction processing support. Components within the executive, such as the file system, and components built on top of the executive, such as a distributed transaction processing system, use these services to achieve atomicity and durability.

The transaction processing services provide the primitives for common logging, generating transaction IDs, recovery of file system and data structures, checkpointing, concurrency control, as well as to begin, end, and abort transactions.

1.7 Process Support

The OZIX executive provides execution support for applications through a process subsystem. The process subsystem is intended to be generic with respect to the API used by the application. For example, the process subsystem can support both a fork/exec model and a create process model, depending on the API being used.

The process subsystem implements process and thread scheduling policy, process signaling, as well as the creation, deletion, and management of processes and threads.

Standard UNIX Interfaces

The following is a list of some of the standard UNIX interfaces supported by the OZIX executive:

- Process Management Services—OZIX supports standard UNIX process management system services, such as fork and exec. The complete list is defined in the Process Management Interfaces section of the *OSF AES Operating System Programming Interfaces System Services Outline, Revision 2.0*. Digital extended COFF is used for the image file format.
- Signal Interfaces— OZIX supports the standard UNIX signal interfaces, such as kill and signal. The complete list is defined in the Signal Interfaces section of the *OSF AES Operating System Programming Interfaces System Services Outline, Revision 2.0*.
- Process Environment Interfaces— OZIX supports the standard UNIX environment interfaces, such as getpid and umask. The complete list is defined in the Signal Interfaces section of the *OSF AES Operating System Programming Interfaces System Services Outline, Revision 2.0*.

Concert Multithread™ Services

The executive supports the Concert Multithread ™ Architecture core services, as defined in Chapter 4 of the *Concert Multithread Architecture, Revision 1.0-2*. This support includes true multithread support in the executive and asynchronous alerts.

Scheduling Features

The executive supports advanced scheduling features to control the scheduling of threads and processes. Possible system options include:

- A fair-share scheduler
- A time share scheduler
- A run to completion with preemption scheduler

1.8 Memory Management Interface

The executive supports the BSD style memory management interface as defined in the Memory Management Interfaces section of the *OSF AES Operating System Programming Interfaces System Services Outline, Revision 2.0*. Mapped files and the data in the file system is automatically kept consistent. This means that, if an application opens a file and issues read/writes while another application has the file mapped, each application can see the changes made by the other.

In addition, memory management supports the following features:

- Maximum sharing of physical memory
- Unlimited number of virtual address spaces (only limited by amount of physical memory)
- Sparse virtual address space
- Multiple page files
- Maximize size of user virtual address space
- Alternate backing store managers and page replacement algorithms
- The ability to use as much physical memory as needed, when physical memory is available
- Fair sharing of physical memory when physical memory is scarce

1.9 I/O Support

The primary goals of the OZIX I/O system are high performance, reliability, and the ability to efficiently manage tens of terabytes of data on hundreds of storage devices. The I/O system supports these goals through the utilization of subsystems, multiple threads, and the Digital *attribute-based allocation* (ABA) architecture.

ABA allows file data to be dynamically mapped to storage devices. This dynamic allocation of data provides the foundation for features such as hierarchical storage management (HSM) and dynamic reconfiguration of storage devices. ABA also provides the basis for disk striping, disk shadowing, volume sets, and logical volumes.

ABA and the OZIX I/O design are discussed further in the I/O System chapter.

1.10 Distributed Computing Environment

OZIX provides a rich distributed computing environment by supporting many defacto network standards. These include the Sun™ Network File System (NFS), a remote procedure call (RPC) facility based on the Apollo® Network Computing System (NCS); distributed naming services via the Berkeley Internet Name Domain (BIND) and Digital's Distributed Name Service (DNS); and authentication services via MIT's Kerberos and Digital Authentication Security Service (DASS).

Network Support

The OZIX network provides the underlying technology for success in a distributed environment. This includes multivendor interoperation via adherence to both defacto and international open standards, a state-of-the-art high performance network implementation, and careful attention to the ability to scale the network to very large configurations.

TCP/IP and DECnet™/OSI

OZIX communication services include both the Internet (TCP/IP) and OSI protocol suites. These protocols provide network interoperability with virtually every major computer vendor. Both are high speed native implementations, accessible to applications through a choice of standardized interfaces, such as Berkeley sockets and the X/OPEN Transport Interface (XTI). Indirect network services, such as remote procedure calls (RPCs) and remote file access, are also provided. All system and network services are managed through a single, common management mechanism.

OZIX adds additional applications and services to provide a wider range of functions to other OZIX, ULTRIX, VMS, and MS/DOS systems. In addition, DECnet/OSI provides gateway access to proprietary IBM SNA networks and CCITT X.25 public data networks.

The OZIX distributed computing environment is discussed in more detail in the Network Implementation Architecture chapter.

1.11 System Administration

OZIX system administration is designed to conform to POSIX, X/OPEN, OSF, and ISO/OSI interface standards, as well as DoD security standards. The OZIX system administration structure is based on object-oriented methodology. All tasks performed by system administration are performed on *manageable objects*, which represent users, devices, files, processes, networks, and so on. As objects, these components are identified by their specific attributes, operations, and events.

Manageable objects are connected through a *management backplane* to each other and to *user presentations*. The management backplane provides a common set of services across interconnected machines. This interconnect is provided by the Enterprise Management Architecture (EMA) and the use of various ISO and IETF network management standards.

A user presentation is a type of *management application* that provides an interface through which system managers manipulate objects on the backplane. Several user presentations are implemented to provide a variety of interfaces for both character-cell and DECwindows terminals. In addition to user presentations, automated management applications are implemented to handle such tasks as system configuration and fault management.

System administration is discussed in more detail in the System Administration chapter.

1.12 Built-in Performance Analysis

The OZIX system is designed to provide built-in performance analysis capabilities through an instrumentation collection subsystem. The instrumentation collection subsystem implements a set of instrumentation services that use collection points in the OZIX code to capture the state of the system at any point in time. Such event collection capabilities provide the information necessary for capacity planning and distributed system debugging. Additional performance information on the OZIX system is provided by performance characterization tools, such as prof, Pixie, and the SA* series of system activity tools.

Performance engineering is discussed further in the Performance Engineering chapter.

1.13 Outline

The following chapters discuss the various OZIX components and implementation strategies in more detail. For complete design information on these topics, refer to the respective functional specifications.

- Chapter 2, User Environment and Application Environment, discusses the OZIX user and application environment. This chapter describes OZIX from the point of view of end users, application programmers, and system administrators.
- Chapter 3, System Administration, describes the overall structure on which the OZIX system administration strategy is based.
- Chapter 4, Resource/Fault Management, discusses how OZIX is to achieve its availability, reliability, configuration, and serviceability goals.
- Chapter 5, Security, discusses the strategies for building the security and integrity into OZIX necessary to achieve a B2 level of certification by the DoD.
- Chapter 6, Internationalization, describes the strategies for building internationalization support into the OZIX system.
- Chapter 7, Base System Architecture, describes the fundamental support mechanisms for the OZIX executive and nub. This chapter discusses how OZIX supports subsystems, manages threads, and implements virtual memory.
- Chapter 8, I/O System, provides an overview of the Digital ABA architecture and discusses the OZIX file system and mass storage designs.
- Chapter 9, Terminal Subsystems, describes how OZIX terminals are supported in the OZIX system.
- Chapter 10, Network Architecture, discusses the OZIX distributed system architecture.
- Chapter 11, System Installation and System directory Layout, describes the OZIX directory structure, as well as the strategies for installing the system software and layered application software.
- Chapter 12, Performance Engineering, discusses the strategies for integrating performance analysis into the basic design of the OZIX system. This chapter also describes how performance instrumentation is to be built into the system for the purposes of performance characterization, system management, capacity planning, transaction processing, and configuration management.
- Chapter 13, Software Quality and Testing Strategy, describes the software testing methodologies used to ensure a quality implementation of the OZIX system.

CHAPTER 2

USER ENVIRONMENT AND APPLICATION ENVIRONMENT

This chapter contains a high-level description of the OZIX V1.0 computing environment as seen by OZIX users. The chapter starts by briefly characterizing OZIX users by type, and then discusses the environment OZIX offers to each of these types of users. For each user environment, the discussion consists of a brief description of the user interfaces and applicable tools as provided on OZIX, any unique OZIX technology used to provide these capabilities, and the major *user-visible* ramifications of this technology.

The purpose of this chapter is to provide a road map to OZIX capabilities from a task- or user-oriented perspective, so as to supplement the component- and subsystem-oriented descriptions that comprise the rest of this technical summary.

2.1 Types of OZIX Users

When discussing the different types of user environments available on OZIX, it is first helpful to divide users into three arbitrary groups.[1]

The first group of users, which we refer to as *end users*, are those users that typically have little contact with anything but the most basic commands and utilities available on OZIX.

The second group of users, which we refer to as *application developers*, are characterized by the software development tools and system routines that they use in addition to the tools used by end users.

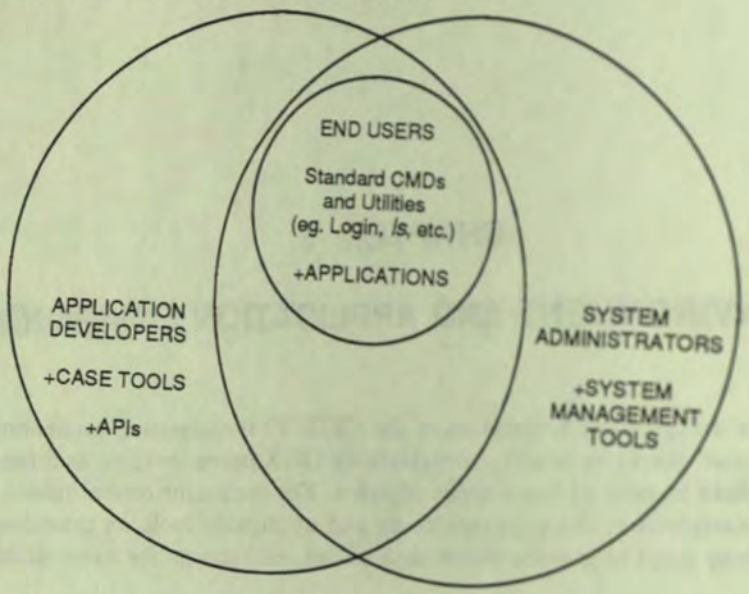
The third group of users, which we refer to as *system administrators*, are characterized by their use of privileged interfaces to OZIX that are used to control the overall function of OZIX.

Figure 2 shows how the user environments of these three groups overlap. This figure is not meant to be rigorous; there are always exceptions to this model. The figure is meant to convey in broad detail that the heart of the OZIX user environment is the end user environment, described in Section 2.3.

In general, all three groups of users use the common commands and utilities contained in the end user environment. Applications developers typically use all of these commands and utilities plus a set of additional utility programs, such as compilers, that are part of the OZIX program development environment. In a somewhat disjoint space, system administrators use many of the end user tools plus a series of system management tools.

[1] This taxonomy of three groups of users is arbitrary. These three groups should be familiar to most readers of this chapter. For a more precise marketing-derived discussion of OZIX users, especially for detailed descriptions of the system administrator user group, see the user model contained in the *OZIX Usability Plan*.

Figure 2: Overlap of the OZIX User Environments



The following three sections describe the main attributes of each type of OZIX user. The rest of this chapter then discusses how all users access OZIX capabilities and goes on to describe each of the OZIX user environments in more detail.

The description of system administrators and their environment contained in this chapter is brief as this information is provided in greater detail in the System Administration chapter and the *OZIX Usability Plan*.

2.1.1 End Users

OZIX end users consist of a broad spectrum of people who use OZIX because their applications run on OZIX. Their key applications run on OZIX because OZIX offers a set of industry standard application program interfaces. End users also benefit from the standard look-and-feel of the OZIX user interface; they bring their experience of working with UNIX, ULTRIX, or OSF/1 systems with them to OZIX.

As a group, OZIX end users are not required to be very computer-literate, but they may be. If they have experience with computers, it is probably based on experience with UNIX, ULTRIX, and OSF/1 systems. Users of turnkey applications may also have experience running on MS-DOS®, VAX/VMS™, or MVS™ systems. In addition to running specific applications, an end user's primary day-to-day tasks include running mail, performing document processing, and performing simple maintenance of their online environment, for example, creating and deleting files. Users of turnkey applications are often aware of only the user environment provided by the application they run.

In general, end users are more interested in getting their job done using the tools available on OZIX, than they are in details of how OZIX works. As a result, they are likely to be unaware of the technical details of many unique OZIX capabilities. Instead, they rely on system administrators to establish default settings for these features for them and then use these features as if they were a "black box." At this level, OZIX enhanced internationalization support and the distributed computing environment is of interest to many end users.

End users use both the character-cell-terminal and workstation interfaces to OZIX; the choice of which type of interface is used is largely based on the economics of per-seat costs of character-cell terminals versus workstations. For users of turnkey applications, the choice of terminal interface is most often determined by the requirements of the applications used.

2.1.2 Application Programmers

OZIX application developers use OZIX because it is based on industry standards. OZIX application developers include "end customers", third-party developers (both ISVs and CMPs), and Digital developers. They develop applications based on OZIX because OZIX offers a set of standard application program interfaces plus Digital added value in the form of a rich set of additional application program interfaces and application development tools. Like end users, application developers also benefit from the standard look-and-feel of the OZIX user interfaces; they bring their experience of working with UNIX, ULTRIX, or OSF/1 systems with them to OZIX.

OZIX application developers are computer-literate; they know how to operate computers at the end-user level and they also know how application programs and operating systems interact. OZIX application developers know how to program and are typically familiar with the C programming language and the application run-time environment of UNIX, ULTRIX, or OSF/1 systems. In addition to performing the same types of tasks that end users perform, application developers design, create, and maintain application programs.

While developing and maintaining their applications, the OZIX application developer's view of the OZIX development environment is similar to the end user's view—using tools as black boxes. Because of their knowledge, application developers are much more demanding about the performance and capabilities of their development environment. Application developers are typically very sophisticated users of shell capabilities; indeed, many application developers use the programmable features of OZIX standard shells as a major supplement to their application code and to customize the development environment. At the level of the development environment, application developers are aware of the presence of OZIX added value and they take advantage of it, for example, by using the additional applications that DECwindows provides. Application developers are also more likely to be aware of capabilities of the OZIX security environment and take advantage of it than are end users.

While designing and upgrading their applications, the OZIX application developer's view of OZIX switches to the level of the application program interface. Application developers make tradeoffs in performance, capability, maintainability, standards-compliance, and so on as an integral part of their job. Application developers take advantage of the presence of added value in OZIX, subject to the tradeoffs mentioned above. For example, the increased internationalization support in OZIX available to applications is of value to application developers.

Application developers use both the character-cell-terminal and workstation interfaces to OZIX; the preferred interface is the workstation interface because of the flexibility that multiple simultaneous windows allow. OZIX application developers are typically very well-versed in the advantages of distributed computing and take advantage of this support from OZIX in both their applications and as part of the OZIX development environment.

2.1.3 System Administrators

The group of users known as OZIX system administrators encompasses a variety of system management and control roles:

- Account Administrator
- Auditor
- Operator
- Security Administrator
- System Programmer
- Network Administrator

Some of the many tasks performed by system administrators include: installing and maintaining the OZIX system and layered products, creating and maintaining user accounts, performing backup and other media management, controlling resources, batch, print, and network control, and controlling the security and integrity features of OZIX,

Because the system administrator role is so broad, it is hard to describe them in general terms. At one end of the spectrum operators are very similar to privileged end users; they use largely fixed operator interfaces to perform a limited range of operations. In this role, system administrators are like turnkey application users. At the other end of the spectrum system programmers are very similar to privileged application developers; they create system-level applications using the same basic environment that application developers use.

OZIX system administrators are computer-literate; they know how to operate computers at a system level. They typically understand the types of resources they control and the implications (in terms of system performance, security, and throughput) of their actions. OZIX added value for system administrators comes in improving the ease-of-use of system administration tools, reducing the level of computer literacy required.

OZIX system administrators are aware of the presence of OZIX added value and the methods used to control it. OZIX' system administration focus is based on a sophisticated window-based user interface that is an obvious improvement over the system management capabilities of other UNIX, ULTRIX, and OSF/1 systems. Control of OZIX' enhanced security and integrity features are of great interest to them, as well. As was mentioned in Section 2.1.1, system administrators are often called upon to establish a usable black-box environment for unsophisticated end users, for example, by establishing the correct internationalization parameters for users.

OZIX system administrators use both the character-cell-terminal and workstation interfaces to OZIX; the preferred interface is the workstation interface because of the existence of OZIX' enhanced system administration support utilities, which require a workstation to run.

For a more detailed description of system administrators see the *OZIX Usability Plan* and the System Administration chapter.

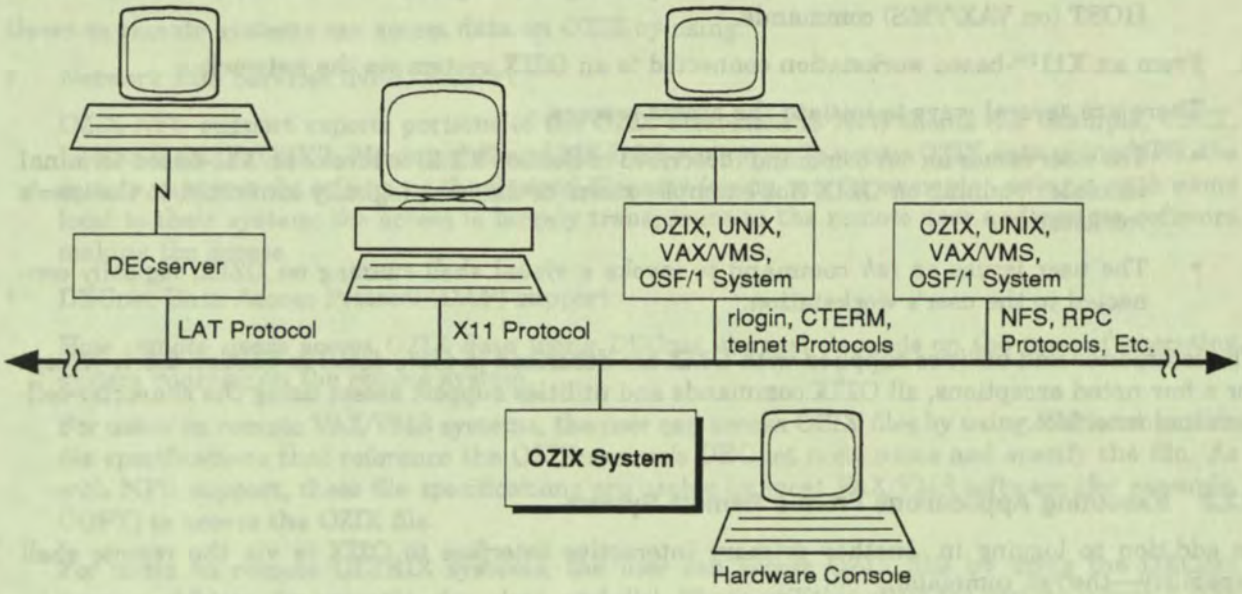
2.2 Ways Users Access OZIX

Like other implementations of OSF/1, users can access an OZIX system by:

- Interactively logging in to OZIX from character-cell terminals or bitmapped terminals/workstations
- Executing applications from a remote system
- Executing applications using the OZIX batch system
- Accessing data that resides on an OZIX system from a remote system

Figure 3 shows a sample networked computing configuration containing an OZIX system, and it shows several examples of the methods for accessing OZIX capabilities. The following sections describe the ways users access OZIX.

Figure 3: Ways Users Access OZIX



2.2.1 Logging in to OZIX

Most users access OZIX via the interactive interface, logging in to OZIX using character-cell terminals or bitmapped terminals/workstations (hereafter referred to simply as workstations).

There are four major ways to log in to OZIX:

1. Locally on a character-cell terminal connected to the system console terminal interface.

   The user initiates the access process in a hardware-specific manner, typically by pressing BREAK on the console terminal.

2. Through a terminal connected to a terminal server on the network.

   The terminal server is typically a LAT server device such as a DECserver™.

   The user initiates the access process by using the LAT server CONNECT command, which logically connects the terminal to the OZIX login processing software.

3. Remotely from a terminal on another system in the network.

   There are several possible remote login methods:

   — The *rlogin* terminal protocol, which can be used from an OZIX, ULTRIX, OSF/1, or other UNIX-style system.

   The user initiates the access process by using the *rlogin* command, which logically connects the terminal to the OZIX login processing software.

   — The *telnet* terminal protocol, which can be used from OZIX, ULTRIX, OSF/1, other UNIX-style systems, or from *telnet* terminal servers, known as "PADs".

   The user initiates the access process by using the *telnet* command, which logically connects the terminal to the OZIX login processing software.

   — The CTERM terminal protocol, which can be used from an OZIX, ULTRIX, or VAX/VMS systems.

The user initiates the access process by using the *dlogin* (on OZIX and ULTRIX) or SET HOST (on VAX/VMS) commands.

4. From an X11™-based workstation connected to an OZIX system via the network.

   There are several ways to initiate the access process:

   - The user issues an *rsh* command (described in Section 2.2.2) to invoke an X11-based terminal emulator running on OZIX (for example, xterm or dxterm), logically connected to the user's workstation.
   - The user issues an *rsh* command to invoke a visual shell running on OZIX, logically connected to the user's workstation.

The commands and utilities supplied with OZIX are described in more detail in Section 2.3.1. Except for a few noted exceptions, all OZIX commands and utilities support access using the character-cell-terminal interface.

2.2.2 Executing Applications From a Remote System

In addition to logging in, another primary interactive interface to OZIX is via the remote shell capability—the *rsh* command.

Users on remote OZIX, ULTRIX, UNIX, and OSF/1 systems can use the *rsh* command to invoke applications, commands, and utilities on an OZIX system.

The OZIX implementation of *rsh* functionality (both incoming and outgoing) is similar to that provided with UNIX, ULTRIX, and OSF/1 systems. For incoming requests, the OZIX Internet daemon receives *rsh* protocol messages sent to the local OZIX system from a remote system. It invokes the *rsh* daemon to execute the specified shell commands, and it performs the pipefitting necessary to ensure that the local *rsh* daemon can communicate with the remote *rsh* code.

2.2.3 Executing Applications Using the Batch System

Users may also access OZIX by submitting jobs to the OZIX batch facility, using the *at* and *batch* commands, as specified by X/OPEN. These commands are issued directly on OZIX, or equivalently, may be executed on OZIX from a remote system through the use of the *rsh* command.

The *at* and *batch* commands buffer shell commands to be executed via the batch facility. Depending on the command used to submit the batch job, processing is performed slightly differently.

Batch jobs submitted using the *at* command are queued by execution time in a data file. The data file is periodically examined by a daemon scheduled via an entry in */etc/crontab*. When a job is found that is due to execute, the daemon forks and executes a shell to process the batch shell script. The daemon performs pipefitting so that, unless redirected by the the batch shell script, the contents of *stdin* and *stdout* are saved and mailed to the submitter upon job completion.

Batch jobs submitted using the *batch* command are queued for immediate execution. When ready for execution, batch jobs are processed in the same way as jobs submitted using the *at* command.

2.2.4 Accessing Data on OZIX From a Remote System

Users on remote systems can access data on OZIX by using:

- Network File Services (NFS) support

  OZIX NFS support exports portions of the OZIX filesystem to NFS clients (for example, UNIX, ULTRIX, OZIX, OSF/1, Macintosh® and MS-DOS systems). To access OZIX data using NFS, the remote user merely references the desired filename (using *cat*, for example), using a path name local to their system; the access is largely transparent to the remote user and remote software making the access.

- DECnet Data Access Protocol (DAP) support

  How remote users access OZIX data using DECnet support depends on the type of operating system running on the remote system.

  For users on remote VAX/VMS systems, the user can access OZIX files by using RMS-compatible file specifications that reference the OZIX system's DECnet node name and specify the file. As with NFS support, these file specifications are usable by most VAX/VMS software (for example, COPY) to access the OZIX file.

  For users on remote ULTRIX systems, the user can access OZIX files by using the DECnet support utilities (for example, *dcp, dcat, and dls*). These utilities allow the remote users to copy the OZIX files to their system for local processing. Unlike NFS support and DECnet support on VAX/VMS, only these support utilities are able to access files on OZIX; all other commands, utilities, and applications are not capable of directly accessing OZIX data.

- ISO File Transfer Access Method (FTAM) support

  How remote users access OZIX data using FTAM support depends on the type of operating system running on the remote system.

  For users on ULTRIX systems (V4.0 and later) and OZIX systems, the *ftam* or *cp* commands can be used to copy data from an OZIX system to the remote system for processing there.

  For users on remote VAX/VMS systems, the COPY command can be used to copy data from an OZIX system to the VAX/VMS system for processing there.

- File Transfer Protocol support

  Remote OZIX, OSF/1, ULTRIX, and UNIX users can copy files from OZIX to their remote system for local processing using the *ftp* command.

- Berkeley file transfer support

  Remote OZIX, OSF/1, ULTRIX, and UNIX users can copy files from OZIX to their remote system for local processing using the *rcp* command.

2.3 The OZIX End User Environment

As shown in Figure 2, the end user environment is the "common ground" for all OZIX users. This common environment is based heavily on standardized software.

The following sections describe the major aspects of the OZIX end user environment:

- Section 2.3.1 describes the standard commands and utilities provided with OZIX that are typically used by end users.
- Section 2.3.2 describes the shells provided with OZIX.

- Section 2.3.3 describes the major features of the OZIX character-cell-terminal user interface.
- Section 2.3.4 describes the major features of the OZIX workstation user interface.
- Section 2.3.5 and Section 2.3.6 describe the user's view of the added value provided with OZIX' enhanced security and internationalization support.

2.3.1 Standard Commands and Utilities

OZIX provides a wealth of commands and utilities whose user interface and function comply with the following formal industry standards and *de facto* industry standards:

- X/OPEN Portability Guide, XSI Commands and Utilities (Issue 3) —an industry standard
- POSIX 1003.2, Shell and Application Utility Interface for Computer Operating System Environments (Draft 8) – an industry standard
- OSF Operating System Component (OSC) Functional Outline: Application Environment Specification (Revision 1.0) – a *de facto* industry standard

In addition, other commands and utilities are provided with OZIX to allow for migration of users from VAX/ULTRIX to OZIX.

End users use a limited set of the total commands and utilities provided with OZIX to perform the following types of basic tasks on OZIX:

- Maintaining files and directories

  Commands used for these types of tasks include: *cat, rm, ls, mkdir, cd, cp, mv, more, view, grep,* and so on.

- Communicating with other users and systems

  Commands used for these types of tasks include: *mail, rcp, talk,* and so on.

- Document processing

  Commands used for these types of tasks include: *vi, spell, nroff, troff, tbl,* and so on.

- Printing documents

  Commands used for these types of tasks include: *lpq, lprm,* and so on.

- Obtaining information

  Commands used for these types of tasks include: *man, who, ps, date,* and so on.

For more information about standard commands and utilities provided on OZIX, see the *OZIX Standard Commands Functional Specification* and the *OZIX ULTRIX-32 Compatibility Commands Functional Specification*. Together, these documents list the names of all standard commands and utilities provided on OZIX.

2.3.2 OZIX Shells

Both character-cell terminal users and workstation users can interface to OZIX using the following shells:

- Bourne Shell—a standard shell
- C Shell—a *de facto* standard shell

These shells provide a simple, line-oriented command interface and function identically to other implementations as provided on UNIX, ULTRIX, and OSF/1 systems. The standardized look and feel of these shells reinforces to users that OZIX is based on standards.

Like other OSF/1-compliant open systems, OZIX allows users to choose a default shell. This shell preference is stored as part of the per-user data managed by the system administration components. Users may also change shells dynamically, as is done on other OSF/1-compliant open systems.

Workstation users may also use a window-based visual shell known as the ULTRIX User Executive. The ULTRIX User Executive provides an icon-based visual user interface to OZIX based on a paradigm of direct manipulation of files to achieve a particular task. The user interface to UUE is common between OZIX, ULTRIX, and VAX/VMS. UUE is not available on non-Digital platforms, such as generic UNIX systems; UUE is part of the added value Digital provides as part of DECwindows.

Users invoke the ULTRIX User Executive in the same way they invoke other window-based applications (see Section 2.3.4.1). Through a customization menu provided by UUE, users can modify and extend various aspects of the function of UUE, including modifying functional defaults (such as default directory used by UUE) and cosmetic defaults (such as the size and color of the windows used by UUE).

In terms of its implementation on OZIX, UUE is just another window-based application that makes calls to the DECwindows API.

2.3.3 Character-Cell-Terminal User Interface

Access to OZIX via the character-cell-terminal interface provides users with a command-line interface that is a superset of the standard command-line interfaces provided on OSF/1, ULTRIX, and UNIX systems. This interface is characterized by the use of a shell that presents a programmable interface to all OSF-specified standard commands and utilities, plus some additional Digital-standard and OZIX-specific commands and utilities.

The character-cell-terminal interface to OZIX is implemented in various components, depending on the type of access involved:

- Access via the console terminal

  The OZIX console port driver and terminal class driver provide the basic software support for this style of access.

- Access via a LAT-based terminal

  The OZIX LAT port driver and terminal class driver coupled to other network software that provides the software support for this style of access.

- Access via the *rlogin*, *telnet*, SET HOST, or *dlogin* commands

  The OZIX PTY port driver and terminal class driver coupled to other network software that provides the software support for this style of access.

2.3.4 Workstation User Interface

Using a character-cell-terminal emulator, workstation users can access OZIX via the character-cell-terminal interface. In addition, workstations users can run X11, OSF/Motif™, and DECwindows applications on OZIX to present a window-style user interface back to their workstation. These applications are hereafter referred to simply as window-based applications.

A set of standard window-based applications is provided with OZIX, including:

- A character-cell-terminal emulator
- A window-based login capability and session manager

OZIX adds value to the standard X11-based and OSF/Motif-based user interfaces in many ways by supporting DECwindows. DECwindows added value includes:

- Additional Digital-supplied window-based applications:
  - — DECwindows MAIL
  - — ULTRIX User Executive
  - — DECwindows OOTB applications, such as the calendar manager.
  - — Digital's DECchart, DECwrite, and DECdecision applications
- Enhanced internationalization support:
  - — Enhanced support for bidirectional and multilingual text, implemented using Digital's compound string technology. For more information about compound strings, see the Internationalization chapter.
  - — Enhanced support for localization of interfaces, implemented using Digital's user interface language (UIL) technology.
- Enhanced application support, implemented using Digital's DECwindows Toolkit enhancements, resulting in a higher degree of consistency in the look-and-feel of application human interfaces.

2.3.4.1 Invoking Window-based Applications

Users can run window-based applications on OZIX with the human interface directed to an X11-compatible workstation on the network. The user starts the application on the OZIX system and specifies the network address of the target workstation.

OZIX provides several ways to start up window-based applications, all similar to mechanisms used on UNIX, ULTRIX, and OSF/1 systems today. These are the primary methods:

- From a workstation, the user can log in to OZIX and issue a command to start up the application. The user logs in using the mechanisms described in Section 2.2.1. The network address of the target workstation can be specified explicitly on the command line, or implicitly through the DISPLAY environment variable.
- From a local login session on a workstation, the user can invoke the *rsh* command to remotely issue a command on OZIX to start up an application. The network address of the target workstation is often explicitly specified as part of the commands specified via the *rsh* command.
- On the OZIX system, the user can submit a batch job (via *at* or *batch*) that starts a window-based application. Alternatively, a system administrator can place an entry in */etc/crontab* to start a window-based application for the user. These techniques can be used to cause the execution of a shell script that periodically starts or stops the execution of window-based applications. The network address of the target workstation is often explicitly specified as part of the commands specified via the *at* command, *batch* command, or the entry in */etc/crontab*.

These are just the most common techniques, others are possible. Most of these techniques rely on using either the standard interactive login support (as described in Section 2.2.1), remote shell support (as described in Section 2.2.2), or batch system support (as described in Section 2.2.3) in some combination to invoke the application.

2.3.5 The End User View of OZIX Security

Security on OZIX can be viewed as a continuum that ranges from systems with minimal security support enabled to systems with full security support enabled. As a result, the end user view of OZIX security varies from system to system, based on the security policy in effect on the system.

On OZIX systems with minimal support enabled, most end users perceive the security aspects of running on OZIX as being largely indistinguishable from other OSF/1-compliant systems:

- The OZIX login dialogue and authentication process (based on user name and password) is very similar to that provided on other OSF/1 systems.
- The concepts of "processes" (on OZIX, executors) having a user identification, group identification, and other basic security information are the same on OZIX as on other OSF/1 systems.
- File protection (rwx) and security characterizations (owner, group, and world) and the commands to manipulate protections are the same on OZIX as on other OSF/1 systems. For example, OZIX users can use the standard *chown* and *chmod* commands to modify file ownership and protection.

Security support provided with OZIX gives system administrators a great deal of flexibility in tailoring the security profile of each individual OZIX system. They can implement their local security policy using the following types of controls:

- Access Control Lists (ACLs)
- Mandatory Access Control (MAC)
- Integrity Access Control (IAC)
- Level control of network access
- Control over access to "raw" devices

In addition, OZIX security support allows the security administrator to change the basic security model in effect on the system. For example, some OZIX systems may use a security model based on the Bell and LaPadula model; other systems may create their own customized security model. There are a variety of other tunable security features provided with OZIX.

As more security features are enabled on OZIX systems, the end user is likely to perceive this change as the direct result of increasing access limitations – information access and exchange becomes subject to more control and restrictions. For many OZIX systems, the OZIX security system works quietly "behind the scenes" and is only visible to end users when an access attempt fails. The access error is reported to most end users via the standard OSF/1 interfaces, for example, via a standard error report of "permission denied' (EACCES) from a standard command or utility.

At the highest level of security support, OZIX systems conform to the National Computer Security Center (NCSC) B2 level of security support.

The following two sections describe two major user-visible aspects of OZIX security: access control lists and the Trusted Computing Base Shell (TCB Shell).

For more information about the concepts and capabilities of the OZIX security system, see the Security chapter, the *OZIX Security Support Component Functional Specification*, and the *OZIX Access Validation Component Functional Specification*.

2.3.5.1 The End User View of Access Control Lists

OZIX systems support the use of ACLs to label files (actually, dataspaces), devices, and shared memory segments with access control information (identifiers plus access type allowed). On many systems, ACLs are established and maintained by system administrators and the use of ACLs is largely invisible to OZIX end users. This type of environment is likely to be found at installations where compatibility with the UNIX end user environment is paramount.

For other systems, and for other types of users such as application developers and system administrators, explicit manipulation of ACLs by users will be more prevalent. Generally, explicit ACL manipulation by these types of users is performed to supplement and enhance the capabilities provided by the basic OSF/1 security model. For example, ACLs can be used to partition the user community with a finer granularity than "owner", "group", and "world."

2.3.5.2 The End User View of the TCB Shell

End users also interact with the Trusted Computing Base Shell (TCB Shell), a user interface to the Access Validation Subsystem (AVSS) and Security Support Subsystem (SSSS). The interaction with these components of the OZIX security system occurs either implicitly or explicitly.

Users implicitly interact with the TCB Shell when they log in to an OZIX system. The TCB Shell intercepts notification of the intention of the user to log in to OZIX. On terminals connected via LAT, the notification of the intention to log in is the result of the CONNECT command. For the console terminal, it is the result of the user pressing BREAK. For other types of access, including *rsh* access, a shell command is used to invoke the TCB Shell.

As part of login processing, the TCB Shell works in concert with AVSS to validate the login attempt. The user is prompted for a login access type \assumed to be logically equivalent to "default/secure-access1/secure-access2, etc."\. If the default access type is selected, requesting a normal OSF/1 login access, the user is prompted for user name and password and logs in as if on any other OSF/1 system. If any access type is selected other than the default, the TCB Shell and AVSS work in concert to perform a more sophisticated login procedure. Examples of these non-default login procedures might involve the use of additional authentication mechanisms such as "smart-cards" or biometrics.

Users explicitly interact with the TCB Shell after they are logged in to OZIX, as the result of pressing BREAK on their terminal. This usage of the TCB Shell is to allow a user to modify security-related aspects of their login session (by creating a new executor with the desired security-related attributes), thereby subjectively "changing their privilege level" within the security universe established by the site security manager. An example of this would be to change the secrecy level of their executor from "TOP-SECRET" to "SECRET".

After pressing BREAK on a logged in terminal, the user is logically disconnected from the terminal session based on the executor in effect at the time. This executor is held "in limbo" until it is either destroyed or resumed as the result of user input to the TCB Shell. The user is then logically connected to the TCB Shell, which offers a menu-based interface with the following capabilities:

- Show Access Class

  Selecting this menu choice causes the current values of the following four security parameters (collectively referred to as access class) to be displayed on the user's terminal:

  — Secrecy Level—for example, "TOP-SECRET"

  — Secrecy Compartments—for example, "US-CIA, NATO"

  — Integrity Level—for example, "SYSTEM"

  — Integrity Compartments—for example, "GENERAL, PRODUCTION, TEST"

For more information on the meaning of these terms and an overview of the OZIX security system, see the Security chapter.

- Set Access Class

  Selecting this menu choice causes the TCB Shell to present another sub-menu, allowing the user to specify values for the access class parameters listed above. Note that the secrecy and integrity labels are specified using a single-valued parameter, chosen from a list of values established by the site security officer. Secrecy and integrity compartment labels are specified as a list of one or more values established by the site security officer. The TCB Shell displays possible labels for each of the access class parameters; it only displays those labels that the user is authorized to use.

- Exit the TCB Shell

  If menu choices made by the user modified the security access class of the executor, the old executor (the one that existed prior to invoking the TCB Shell) is destroyed[1] and the new executor is created with the new access class, running the user's default shell. The user is logically connected to the new executor; the effect is as if the user logged in again.

  Otherwise, if no modification of the access class of the executor was requested (for example, if access class information was only displayed and not modified), the user is logically reconnected with the old executor and continues the interrupted session as if the interaction with the TCB Shell had not occurred.

2.3.6 The End User View of OZIX Internationalization Facilities

The enhanced internationalization support available in OZIX is visible as part of the end user environment in three ways:

- Enhanced support for non-ASCII text and international data provided by the standard commands and utilities.
- Enhanced support for non-ASCII terminals and locale-specific terminal input processing provided by the terminal class driver.
- The mechanisms used to control these enhanced capabilities.

The following sections discuss each of these topics in turn.

2.3.6.1 Internationalization and Standard Commands and Utilities

OZIX adds value to the standard commands and utilities mainly in the area of increased internationalization support. With respect to internationalization support, there are three types of OZIX commands and utilities:

- Commands and utilities that are functionally equivalent to the existing standard-compliant commands and utilities available on UNIX, ULTRIX, and OSF/1 systems.

[1] It is destroyed because it is not possible to change the security access class of an executor. Although this chapter talks loosely of changing or modifying the security access profile of an executor, and that is largely the effect the user sees when using the TCB Shell, the actual mechanism is to create a new executor and destroy the old one. It is undecided if the TCB Shell will also allow "parking" of the old executors instead of destroying them; this would allow users to have access to several executors, accessible one at a time.

The OZIX versions of these commands and utilities perform equivalently with other implementations on UNIX, ULTRIX, and OSF/1 systems. How well each command and utility supports execution in a non-English, non-ASCII environment depends on which standard the command is compliant with; X/OPEN XPG3-compliant software performs the best with respect to internationalization support.

In general,these commands and utilities may not process non-ASCII input correctly. They may not be able to support locale-specific processing of dates, currency, collation order, bidirectional text, multiple languages, and so on.

- OZIX versions of selected commands and utilities that have been made "8-bit clean."

  The OZIX versions of these commands are modified so that they will not arbitrarily remove the high-order bit of each input byte they process. Generally, this improves the ability of these commands and utilities to minimally process non-ASCII text without error. Like the unmodified commands and utilities mentioned in the previous category, these commands may still be severely limited in their support of processing international data.

- OZIX versions of selected commands and utilities that have been fully internationalized.

  The OZIX versions of these commands and utilities use Digital added value in the form of Digital's extensions to X/OPEN's Natural Language Support (NLS) and use OZIX' compound string API support to support processing multibyte, multilingual, and bidirectional text data. These commands and utilities also allow a user to specify preferences for locale-specific processing of dates, currency, collation order, bidirectional text, multiple languages, and so on.

2.3.6.2 Internationalization and Terminal Driver Support

OZIX also adds value to the OSF-specified character-cell-terminal interface by adding extensive support for character-cell terminals that use character code sets other than ASCII or the Digital Multinational Character Set (MCS). In addition to supporting ASCII- and MCS-based terminals, OZIX fully supports the use of many non-ASCII/non-MCS terminals such as Digital's VT382 terminal (used to support Asian markets).

The level of internationalized terminal support available on any single terminal connected to an OZIX system also depends on the level of hardware support for transmitting/receiving arbitrary character codes over that hardware. For example, the console terminal interfaces available on many OZIX hardware platforms are often limited in terms of supporting non-ASCII/non-MCS character set data. Accordingly, internationalized terminal capabilities are typically reduced when accessing OZIX via the console terminal on these hardware platforms.

The OZIX terminal class driver offers customizable locale-specific terminal processing support for input preprocessing and output postprocessing. An example of input preprocessing would be support for Japanese-language input preprocessing; this capability allows for user-customized input of Japanese using several, equivalent representations of a word. An example of output postprocessing would be in the conversion of an internal processing codeset representation of output, to a terminal-specific codeset. The terminal driver allows for the dynamic insertion of these locale-specific processing modules.

For more information about the OZIX terminal drivers, see the I/O System chapter and the *OZIX Terminal Subsystem Structural Overview*.

2.3.6.3 How End Users Control Internationalization Support

Most aspects of OZIX internationalization support are affected through the use of the *setenv* command. The *setenv* command is used to specify environment values used for modifying the behavior of the NLS and messaging components of OZIX. This method for controlling internationalization support is specified as part of the X/OPEN Portability Guide.

To control the internationalization support provided by the OZIX terminal driver, a slightly different approach is used. A human user would typically be unaware of the mechanism used to specify the desired international terminal support, as it is most commonly set as part of the login process, based on the user's profile. Users can dynamically change the terminal class driver's processing behavior, for example to change input preprocessing, as in Japanese terminal support.

2.4 The OZIX Application Development Environment

The following sections describe the two halves of the OZIX application development environment:

- Section 2.4.1 describes the OZIX program development environment: the application developer's view of OZIX.
- Section 2.4.2 describes the OZIX run-time execution environment: the application's view of OZIX.

2.4.1 OZIX Programming Tools

OZIX programming tools are based on the use of standard commands and utilities (described in Section 2.4.1.1) as specified by X/OPEN, OSF, and POSIX. As such, the look and feel of OZIX programming tools are mostly identical to those provided on other UNIX, ULTRIX, and OSF/1 systems. Section 2.4.1.2 describes some of the Digital added value provided by the OZIX programming tools.

2.4.1.1 Standard Commands and Utilities

OZIX application developers create and maintain applications using a variety of programming languages such as:

- Ada™
- BASIC
- C
- C++
- COBOL
- FORTRAN
- Pascal

Application developers use all of the commands and utilities used by end users (see Section 2.3.1) plus additional commands and utilities to perform the following types of tasks on OZIX:

- Create and compile source code

  Commands used for these types of tasks include: *cc, cb, lint, lex, yacc, gencat*, and so on.

- Link and debug applications

  Commands used for these types of tasks include: *ld, dbx*, and so on.

- Analyze application performance

Commands used for these types of tasks include: *pixie, prof*, and so on.

- Manage the program development process

  Commands used for these types of tasks include: *make, sccs*, and so on.

- Create utility programs and sophisticated shell scripts

  Commands used for these types of tasks include: *grep, find, awk, sed, echo, tee*, and so on.

- Distribute applications using remote procedure call (RPC) technology

  The *nidl* command is used to invoke the Network Interface Description Language (NIDL) compiler used to convert RPC entry point definitions into C source code.

2.4.1.2 Digital Added Value Commands and Utilities

For the most part, the presence of Digital added value in the OZIX program development environment is the direct result of added value in the application run-time environment (see Section 2.4.2.4). OZIX adds value to the standard program development environment by:

- Enhancing the linker (*ld*) to create shared libraries. Several new *ld* command options have been added to support this. For more information, see the *OZIX Linker (ld) Functional Specification*.
- Adding support for building and loading subsystems.
- Adding support for building secure applications.
- Adding support for generating enhanced message catalogs. This support allows for the storage of message text in compound string format.

The look and feel of these additional commands and utilities (or extensions to existing standard-specified commands and utilities) is patterned after the UNIX-style command line interface style specified by X/OPEN, POSIX 1003.2, and OSF.

For workstation users, DECwindows added value includes:

- The User Interface Language (UIL) Compiler

The look and feel of the DECwindows tools is consistent with the *DECwindows Style Guide*.

2.4.1.3 Third-Party Tools

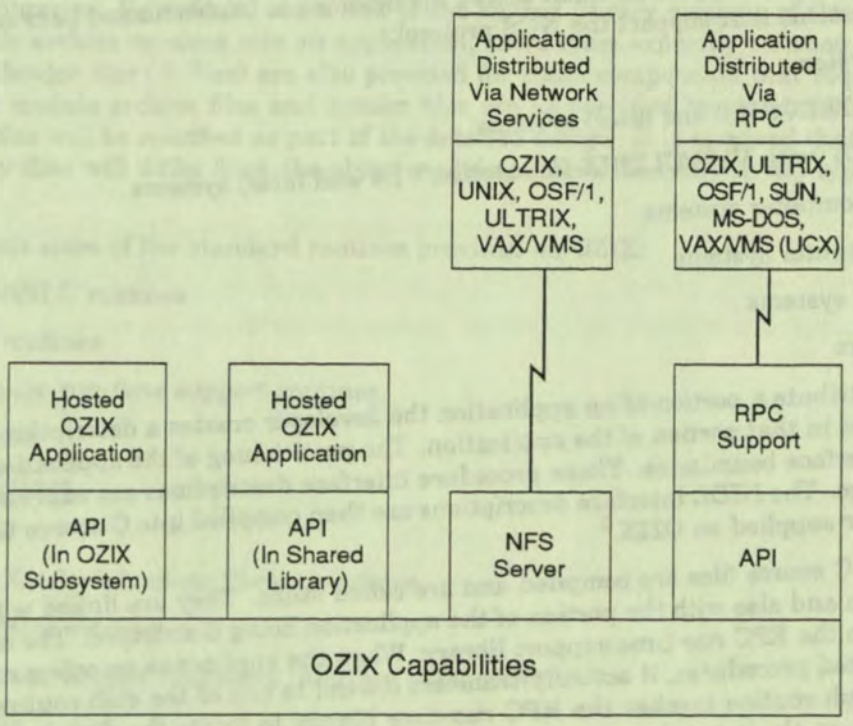
OZIX will support the execution of the INGRES™ relational database product, including Structured Query Language (SQL) support.

For more information about third-party tools on OZIX, see the *OZIX Applications Plan*.

2.4.2 The Application Execution Environment

Figure 4 shows a sample OZIX computing environment in which four applications are running.

Figure 4: Ways Applications Access OZIX



The figure gives four examples of the three basic ways applications access OZIX capabilities:

- Executing directly on OZIX

  The leftmost pair of applications shows the most common way applications access OZIX capabilities—by executing directly on OZIX and invoking documented application program interfaces. OZIX supports all of the common mechanisms for packaging API components: source files (.h files), object files (.o files), object module archive files (.a files), and so on.

  The leftmost two applications in Figure 4 also show two additional interface packaging techniques that OZIX adds to the list above. One application references an API implemented using an OZIX subsystem; the other application references an API implemented using a shared library. Section 2.4.2.4 describes the unique features of OZIX subsystems and shared libraries.

- Accessing OZIX via a network server interface[1]

  The third example in Figure 4 shows an application that accesses OZIX via a network server interface. Examples of this type of application access supported on OZIX are Network File System (NFS) server support and DECnet Data Access Protocol (DAP) support.

- Invoking OZIX capabilities via RPC

  The fourth example in Figure 4 shows an application that accesses OZIX via a specialized form of network access support—remote procedure call.

[1] The orientation of this chapter is on other systems accessing OZIX systems. It should not be overlooked that OZIX systems also originate many types of network requests.

OZIX includes support for Digital's DECrpc V1.0. DECrpc is Digital's version of HP/Apollo Computer's *de facto* industry standard RPC architecture incorporated in their Network Computing System (NCS). RPC support on OZIX allows applications to be distributed between OZIX and a variety of systems that support the NCS protocol:

— OZIX systems

— ULTRIX (V4.0 and later) systems

— VAX/VMS (with VMS/ULTRIX Connection V1.4 and later) systems

— Apollo Computer systems

— Sun Computer systems

— MS-DOS systems

— and others

Briefly, to distribute a portion of an application the developer creates a description of the procedure interfaces in that portion of the application. The partitioning of the application is based on procedure interface boundaries. These procedure interface descriptions are expressed using the NIDL language. The NIDL interface descriptions are then compiled into C source files using the NIDL compiler supplied on OZIX.[2]

The resulting C source files are compiled and are called stubs. They are linked with the application program and also with the portion of the application being distributed. The stubs provide the interface to the RPC run-time support library. When the application transfers control to one of the distributed procedures, it actually transfers control to one of the stub routines described above. This stub routine invokes the RPC run-time library to locate the distributed portion of the application on the network and to connect to the stub routines contained in it. The remote stub routines convert RPC protocol messages into the actions required to execute the distributed procedure code and return any outputs back to the original caller.

For more information about RPC on OZIX, see the Network Implementation Architecture chapter and the *OZIX Remote Procedure Call Functional Specification*.

2.4.2.1 Standard Application Program Interfaces

OZIX provides a set of standard entry points that are a superset of OSF/1 systems, based on the following overlapping set of documents:

- OSF Operating System Component Functional Outline— Open Software Foundation™
- OSF Application Environment Specification—Operating System Component— System Service Outline, Revision 2.0
- ANSI X3J11 for Programming Language C
- POSIX 1003.1 Portable Operating System Interface for Computer Environments
- X/OPEN Portability Guide, Issue 3, Volume 2, System Interfaces and Headers
- X/OPEN Portability Guide, Issue 3, Volume 7, Network Services
- Federal Information Processing Standard (FIPS) Publication 151

[2] This example assumes that the RPC is originating on an OZIX system. For a more rigorous and complete description of the process of using RPC, see the *OZIX Remote Procedure Call Functional Specification*.

The binary code for these APIs is provided on OZIX in two forms: as object module archive files and also as shared libraries. By default, users link to the shared library versions of these APIs; to link the object module archive versions into an application, users must explicitly mention the archive file when linking. Header files (.h files) are also provided for those components that require them. File names of object module archive files and header files are as specified by standards. File names of shared library files will be specified as part of the detailed design. It is assumed that the file names of shared library files will differ from the object module archive files only in file type (for example, *.sl*).

The following lists some of the standard routines provided on OZIX:

- Standard ANSI C routines
- FORTRAN routines
- Other language run-time support routines
- Math routines
- Termcap routines
- Plot routines
- Curses and X11-based curses library routines
- Primitive DBMS routines
- BSD Socket and X/open Transport Interface (XTI) routines
- GKS routines
- PHIGS routines
- X11 Toolkit and widget support routines
- OSF/Motif support routines

Even though the functional behavior of these routines is specified by standards, the OZIX implementations of *selected* routines adds value by making these routines "8-bit clean" (see Section 2.3.6.1 for a description of the benefits of 8-bit clean code) or by making these selected routines safe for use in multithreaded applications ("thread safe").

For more information, including a full listing of the system calls and library routines supplied with OZIX, see the *OZIX Standard C Libraries Functional Specification* and the *OZIX Miscellaneous ULTRIX Libraries Functional Specification*.

2.4.2.2 Digital Added-Value Application Program Interfaces

OZIX adds value to the application program run-time environment by offering:

- Improved functionality over other OSF/1 systems
- Improved robustness, integrity, and security over other OSF/1 systems
- Improved compatibility or interoperability with other systems

The following lists examples of OZIX added value in the application run-time environment:

- DECwindows Toolkit routines— more functional than X11, OSF/Motif routines
- Compound Document Architecture DDIF support routines – more functional than other OSF/1 offerings

This capability allows an application to store, retrieve, and format message text stored in compound string format.

For more information about compound string support and internationalization support on OZIX, see the Internationalization chapter.

- Security support technology

  OZIX enhanced security technology is available to OZIX application developers (including third-party application developers) in several forms by:

  — Providing APIs used for the development of secure applications.

  The application interface to the security system is provided by the OZIX Security Support component routines. These routines perform auditing, quota management, and covert channel control. See the *OZIX Security Support Component Functional Specification* for more information.

  — Allowing third-parties to install new access control models – these include security, integrity, discretionary, trustedness, or entirely new models.

  — Allowing third-parties to install new access control classes.

  — Allowing third-parties to install new security authenticators.

- Common language calling standard technology

  OZIX provides common interlanguage calling technology with the following features:

  — Language interoperability

  — Parallel multithread application execution

  — Reliable exception handling, including multilanguage uplevel-GOTO support

2.5 OZIX System Administrator Environment

In addition to the commands and utilities used by end users, OZIX system administrators use specialized system administration tools to control the operation of OZIX systems. Section 2.5.1 describes the system administrator environment provided by the standard system administrator commands and utilities. Section 2.5.2 describes the added value OZIX provides in the system administrator environment.

For more information about OZIX system administration, see the *OZIX System Administration Overview* and the System Administration chapter.

2.5.1 Standard System Administration Tools

The system administrator environment as specified by OSF is based on a series of individual commands and utilities. These commands and utilities each have a single function and support a limited character-cell-terminal user interface. Examples of these commands and utilities are: *adduser, removeuser, df, ps*, and *kill*.

Although these commands and utilities are implemented using the OZIX management backplane (described in Section 2.5.2), their function and user interface conform to standards, with a few OZIX-specific extensions.

A complete list of the standard system administration commands and utilities may be found in the *OZIX System Administration Functional Specification*, *OZIX Standard Commands Functional Specification*, and *OZIX Compatibility Commands Functional Specification*.

2.5.2 OZIX Added Value

The heart of OZIX added value in the system administrator environment is the object-oriented technology of OZIX system administration, which embodies Enterprise Management Architecture (EMA) concepts.

OZIX system administration provides a conceptual framework in which system administration activities across a wide variety of tasks, including user access management, security management, storage management, network management, resource management, and configuration management can all be described in a consistent fashion. The OZIX system administration paradigm is based on a universe in which management applications use the management backplane to perform management operations on manageable objects. As a result of this simple paradigm, the OZIX system administrator environment provides a single, unified management control center for all of the types of tasks listed above.

The OZIX management backplane provides a common software interconnect allowing centralized system administration across a variety of networked systems, including OZIX and ULTRIX systems. As it is based on ISO/OSI standards, the OZIX management backplane can be extended to manage systems provided by other vendors.

For more information about the OZIX management backplane, see the System Administration chapter and the *OZIX Management Services Functional Specification*.

In addition to the standard system administration commands and utilities described in Section 2.5.1, OZIX provides three added value system management tools:

- The Management Command Language (MCL) management tool

  This tool provides a character-cell-terminal user interface to manage any manageable object on OZIX in an object-independent fashion. MCL is an extension to the DECnet Network Control Language. MCL is invoked using the *mcl* command and is used for managing objects not managed by the standard commands and utilities and for use with command scripts. This tool is also used in a B2 secure environment to provide a certifiable, trusted path from the system administrator to the managed object.

- The generic object manager

  This tool provides a window-based workstation user interface to an object-independent general management tool. Invoked by the *mcl -d* command, this tool is functionally equivalent to the MCL tool described above. The only difference between the two tools is in the user interface.

- The storage manager

  This tool provides a window-based workstation user interface to a specialized tool used to configure and control aspects of OZIX relating to the Attribute Based Allocation (ABA) storage management technology. This tool presents a sophisticated user interface to manipulate containers, devices, files, dataspaces and other ABA-related objects.

The character-cell-terminal-style tools described above all employ a standard verb/subject-type of command line interface. The window-based tools use the visual paradigm of direct manipulation of objects that is common among window-based applications.

Another added value tool provided in the OZIX system administration environment is the Online Diagnostic Monitor (ODM). ODM provides a character-cell-terminal interface used to control the execution of online diagnostic programs. ODM is also used to control the execution of the Online System Exerciser (OX). For more information about ODM and OX, see the Software Quality and Testing Strategy chapter, the *Online Diagnostic Monitor Functional Specification*, and the *Online System Exerciser Functional Specification*.

2.6 Summary

OZIX is based on standards: user interface standards, command and utility program standards, and application program interface standards. Within this framework of standards, OZIX provides several unique enhancements: an improved user interface, flexible security support, improved internationalization support, improved system administration capabilities, and basic technology improvements over other OSF/1 systems.

This combination of industry standards and Digital added value result in a truly flexible and powerful system suitable for a wide spectrum of applications and users.

CHAPTER 3

SYSTEM ADMINISTRATION

3.1 Introduction

This chapter describes the system adminstration of OZIX Version 1.0 . This chapter introduces the reader to the problems with system administration and the goals of OZIX system administration. This chapter explains the three major components of OZIX system administration: the manageable object, the management backplane, and the user presentation. Interoperability is discussed, followed by a list of system administration components and references to related documents.

3.2 Problem Summary

System administration historically has been developed in an as-needed fashion. When new functions have been invented, new methods and tools have also been invented to manage those functions, often as an afterthought. Any relationship that the new function has with the rest of the system is often left to the system administrator to resolve. An overall framework for system administration has never been defined.

The state of system administration today is poor:

- Production systems are extremely complex and expensive to manage
- Non-production systems are frequently severely limited in their management functionality
- Both production and non-production systems suffer from a preponderance of ad-hoc facilities and interfaces and lack support for distributed and network environments
- No two heterogeneous systems can be managed as a whole.

Solving the system administration problem is a non-trivial task. It requires multiple generations of product offerings as stated in the *OS/SB Enterprise Management Approach: An Overview*. The solution also requires coordinated activity across a large number of groups both internal and external to Digital. This chapter describes the overall structure or framework for system administration in OZIX Version 1.0 . This structure is a foundation on which solutions can be built and is a first step in the evolution of system administration.

3.3 Goals

The goals of OZIX system administration Version 1.0 are as follows:

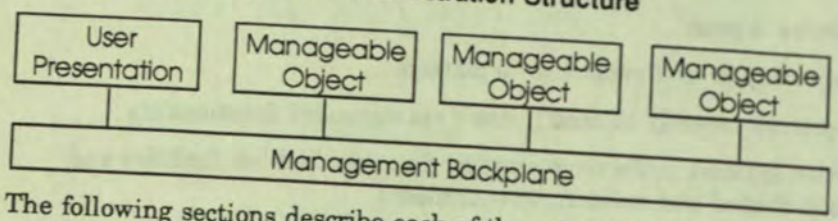
- Provide the first step to managing a network of systems. System administration is not limited to a single node.
- Reduce the cost of system administration of OZIX systems participating in a local area network (LAN).

- Integrate management tasks. Management is performed based on administrative task instead of operating system feature.
- Provide for consistent management. Both philosophy and interfaces for management are consistent across all objects to manage.
- Conform to the emerging Enterprise Management Architecture (EMA). EMA, using the International Organisation for Standardisation's (ISO) Open System Interconnection (OSI), is a definition of a framework for management of heterogeneous, multi-vendor, distributed computing environments and the communications facilities that link them together. It is a Digital corporate-wide architecture.
- Provide adherence and compatibility with Open Software Foundation (OSF), Institute Of Electrical and Electronics Engineers, Inc.'s (IEEE) Standard Portable Operating System Interface for Computer Environments (POSIX), and the X/OPEN Company Limited's X/OPEN Portability Guide. The *OZIX Product and Market Requirements Document* contains the priority order and conflict resolution guidelines.
- Provide management of a production-quality, mass storage system.
- Provide administration of an OZIX system that meets Department of Defense security class B2 rating.

3.4 OZIX System Administration Structure

The structure of system administration is based on object-oriented methodology. The basic building block of system administration is the manageable object. *Manageable objects* are interconnected through a *management backplane* to each other and to *user presentations*. This basic concept is illustrated in Figure 5.

Figure 5: Basic System Administration Structure



The following sections describe each of these in more detail.

3.4.1 Manageable Object

Object-oriented design is a method in which the design is based more on the objects than on the operations performed. The invocation of operations on objects is consistent across all objects. There is much text written on the subject of object-oriented design and programming, and the reader is assumed to have a basic knowledge of the concepts. Because there is no standard terminology used in the object oriented design world, this chapter defines the terminology used in OZIX system administration.

The fundamental building block of OZIX system administration is the manageable object. All tasks required by system administration are performed using the basic structure called a manageable object. Manageable objects are registered with the management backplane. Note that the term manageable object does not relate to the term object as used by the OZIX base system. When referring to manageable objects this chapter simply uses the term object.

For the sake of clarity, a file will be used as an example of an object through this discussion. An object consists of the following basic concepts:

- *Object class*

  Object class is the definition of the object. It is the abstract data structure or template for the object. The object class definition consist of attributes, operations, and events. The object class is analogous to a record or data structure definition in a programming language. An object class has a name which uniquely identifies it from all other object classes.

  A file object class is defined as follows:

Class Name	Attributes	Operations	Events
File	Name	Create	Modified
	Size	Delete	Access Failure
	Creation Date	Set	
	Owner	Get	
	Protection	Backup	

- *Object instance*

  The object instance refers to the actual object. It is sometimes referred to as the instantiation of the object class. The object instance is analogous to the actual record in a data file. An object instance has a name which uniquely identifies it from all other object instances within its object class.

  A file object instance might be as follows:

Attribute	Value
Name	myfile.dat
Size	15
Creation Date	26 October 1989
Owner	JohnDoe
Protection	MaryPublic=read JohnPrivate=write

- *Attributes*

  Attributes are internal state variables of the object, which are made visible across the management interface. The object class defines these attributes. Examples of attributes are identifiers, status, counters, and characteristics. In an object instance, each attribute takes on a value and may be examined and modified by an operation on that object instance. In the file object example above, the object class defines the attributes for the file. The object instance of the file object then has values associated with those attributes.

- *Operations*

  Operations are performed on object instances. The object class defines the operations. An operation is initiated outside of the object on which the operation is performed. For the purpose of system administration, operations are performed on object instances, including the *create* operation (the *create* operation is sometimes considered an operation on the object class; however, for consistency, system administration considers it an operation on the instance. This is more of a theoretical difference, as some code somewhere creates an instance of an object).

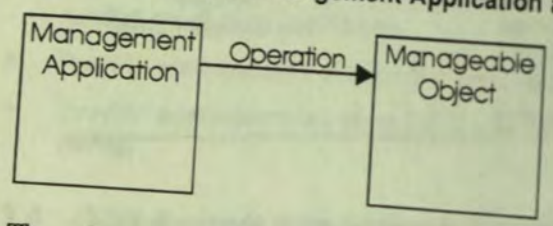
- *Events*

  Objects report events. Events are asynchronous occurrences of normal and abnormal conditions within an object and are triggered by a state transition within the object. For the file object example, a possible event would be an access failure. Access failures are detected when a program tries to open files for which the program does not have access. If access failure occurs, not only would the program be returned a failure status, but the design of the file object may dictate that an event is also reported. The file object does not care who receives the event, only that it must generate the event. In this case, a security monitor may receive the event.

Management operations are performed on objects. This implies that something can request an operation on an object. The requester of an operation is referred to as the *management application*. The relationship between the object and the management application is illustrated in Figures 6 and 7.

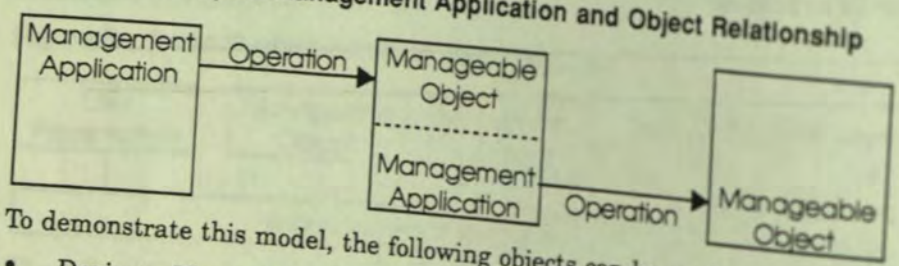
Figure 6 illustrates a simple management application and object relationship. The management application initiates an operation. The object receives the operation and acts upon it.

Figure 6: Simple Management Application and Object Relationship



The roles of management application and object are not static. A management application can in turn be a manageable object if the application requires management. Figure 7 illustrates a more complex set of relationships.

Figure 7: Complex Management Application and Object Relationship



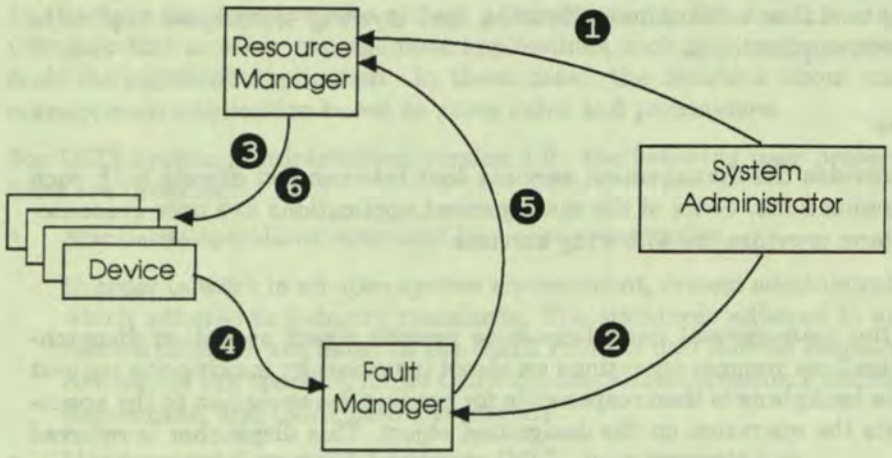
To demonstrate this model, the following objects can be used:

- Devices - These objects represent the devices attached to the system. They have attributes such as on/off line that may be modified with the *set* operation. Device objects also report events when they have errors.
- Fault Manager - This is a management application that monitors the number of soft errors on devices. The fault manager decides, based on error frequency, type, and trends, when a device is failing. It is also an object that is managed. It has attributes that may be modified with a *set* operation, for example, acceptable error thresholds.
- Resource Manager - This is a management application that takes devices online and offline. It is responsible for migrating devices in and out of shadow sets. It is also an object that is managed and has attributes that are modified with a *set* operation. Such attributes include a list of spare devices and the number of replicated devices required in a shadow set.

- System Administrator - The System Administrator uses a management application which is a user presentation. This management application allows the system administrator to communicate to the other objects on the system.

Figure 8 illustrates the objects and management applications described above and some of the relationships and operations that take place. Note that this example is not necessarily the specific model being used in the OZIX product for device fault management.

Figure 8: Modeling Example



❶ The system administrator, using the system administrator user interface (management application), issues the *set* operation on the resource manager (manageable object) to establish the number of replicated devices in each shadow set and to create a list of spare devices.

❷ The system administrator, also using the system administrator user interface (management application), issues the *set* operation on the fault manager (manageable object) to establish error thresholds that the administrator feels his or her computing environment can tolerate.

❸ As a result of ❶, the resource manager establishes the system's resource configuration. This is done by the resource manager (management application) issuing the *set* operation on the device objects (manageable objects) to bring devices online.

❹ As the system runs, the device object (manageable object) reports errors as events. The device object is not concerned with who receives the events; this detail is handled by the management backplane (explained in Section 3.4.2, Management Backplane). In this example, the fault manager (management application) has requested that it receive all error events from the device objects. As the device object reports events, the fault manager receives them.

❺ Using the threshold parameters established by the system administrator in ❷, the fault manager determines that a device is passed the tolerable level of soft errors. The fault manager (manageable object) decides that a device should be removed and reports an event stating the decision. The fault manager is unaware of who receives the event. The receiver might be an automated resource manager (management application) that automatically replaces the device. Alternatively, the receiver might be an operator console (management application), in which case device replacement is a manual operation. The former is the case for this example.

❻ After being notified that a device should no longer be used, the resource manager makes a determination regarding whether the device should be replaced with one from the spare list. If so, the resource manager (management application) issues the *set* operation to the new device

(manageable object) to bring the device online as a member of the shadow set, migrates the data from the old device to the new, and then removes the old device from the shadow set.

Modeling the system in this manner promotes a modular design of system administration. Objects and management applications can be replaced without modification of other objects and management applications on the system. For example, if a resource manager is developed that provides a different replacement algorithm, the resource manager can be replaced on the system without modification to the device, fault management, and even possibly the user interface. The same strategy used to create a piece of code using modules, subroutines, libraries, and layering techniques applies to creating objects and management applications.

3.4.2 Management Backplane

The management backplane provides the management services that interconnect objects with each other and with management applications. Some of the management applications are user presentations. The management backplane provides the following services:

- Operation dispatching

  The primary function of the management backplane is to provide object operation dispatching. All management applications request operations on object instances by making the request through the backplane. The backplane is then responsible for passing the operation to the appropriate code that implements the operation on the designated object. This dispatcher is referred to as the *agent*.

- Event services

  Objects report events to the management backplane. The management backplane receives the events through an *event dispatcher* and forwards the event to the appropriate *event sink*. In order to receive events, management applications interested in specific events subscribe to the appropriate event sink in the management backplane.

- Class definitions repository

  The management backplane provides a repository of object class definitions. An object class definition includes those items described in Section 3.4.1, Manageable Object. This allows other objects and management applications to query the management backplane for a description of an object class. Management applications need not have hard-coded details about an object class, but instead, can dynamically retrieve the definition of the object class. This repository is referred to as the Management Information Repository (MIR).

The management backplane also provides the above services across interconnected machines. This implies that a set of machines has a single, interconnected backplane. This interconnect is provided through such architectures as EMA and the use of ISO/OSI network management standards. One of the many benefits this provides is that a user presentation on one machine will be able to manage objects on another machine.

A higher-level architecture for the management backplane can be found in the *OS/SB Enterprise Management Approach: An Overview*.

3.4.3 User Presentation

The user presentation is the system administrator's window to the management of the system. It is a type of management application, and therefore connects to the management backplane.

The decision making that occurs at the user presentation application is not in the application itself, but in the system administrator using the interface. The intelligence that is built into the user presentation is only to make the interface more user-friendly and aid the system administrator in decision-making.

In the user presentation, the system administrator is the decision maker or at least the observer. Compare this to other management applications such as a resource management application and a fault management application. In these cases, the decisions about management are made by the management application based on some rules and parameters.

For OZIX system administration version 1.0 , the following user presentation management applications are provided:

- Standards-compliant command line user presentation

  In order to work in an open system environment, system administration provides a command line which adheres to industry standards. The standards adhered to and the priority order used to resolve conflicts are listed in the *OZIX Product and Market Requirements Document*. The exact commands are specified in the *OZIX System Administration Functional Specification*, *OZIX OSF Commands*, and *OZIX BSD Commands*.

- Management Command Language (MCL) user presentation

  The MCL user presentation is an extension of the DECnet Network Control Language (NCL) utility. MCL allows management access to any manageable object (assuming normal security checks). MCL has no object-dependent, built-in information or intelligence. It only has knowledge of the object class definition of each object: attributes, operations, and events. MCL is used for managing objects not managed by standards compliant command-line user presentations or the DECwindows user presentations described below. Additionally, MCL is used for debugging, maintenance purposes, or command script processing.

  For certification under B2, a trusted path must be provided from the system administrator to the object being administered when performing security-related management. For this reason, a trusted user presentation is provided, which is a subset of the MCL interface.

- DECwindows user presentations

  DECwindows user presentations are the key to added-value in system administration for the OZIX product. There are two DECwindows presentations provided with OZIX: a generic object manager and a specialized manager.

  The generic object manager is a DECwindows equivalent to MCL. This user presentation manages any object in a generic fashion. As with MCL, no object dependent knowledge is in this DECwindows user presentation.

  True added-value comes with providing specialized, task-based, DECwindows user presentation. The specialized DECwindows user presentation for OZIX Version 1.0 system administration highlights the emerging Attribute Based Allocation (ABA) storage management technology.

3.5 Interoperability

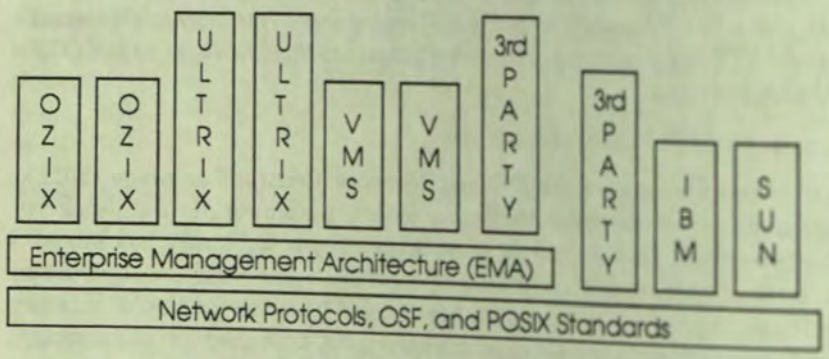
If OZIX system administration creates the "best" system administration system and ignores interoperability with other systems and applications, system administration has failed. OZIX system administration must interoperate as defined below:

- Interoperability within the system, or better put, *intra*operability. The system must be administrated as a system. Failure to achieve intraoperability precludes interoperability in any form. Intraoperability is achieved by cooperation and review among components of OZIX during the specification and design of the system.

- Interoperability with other systems in a network. This includes both OZIX systems, other Digital systems, including ULTRIX and VMS, OSF-compliant systems, and other vendor ISO-compliant operating systems.

  Interoperability with other systems is achieved on two planes: technical and conceptual. Figure 9 shows the OZIX strategy for achieving technical interoperability with other systems. Important tools in achieving interoperability are Digital's EMA, ISO/OSI management standards and protocols, and adherence to other standards such as OSF and POSIX.

Figure 9: Technical Interoperability



  Conceptual interoperability of system administration is achieved largely through common object definitions. For example, the basic concept of files is similar on all machines. Common object definition also allows for variances such as different file name syntax.

- Interoperability with applications that build on top OZIX. These include Digital-supplied and third-party-supplied applications. Easy incorporation of these applications into the management framework allows the system administrator to manage these applications as part of the system using the same methodology.

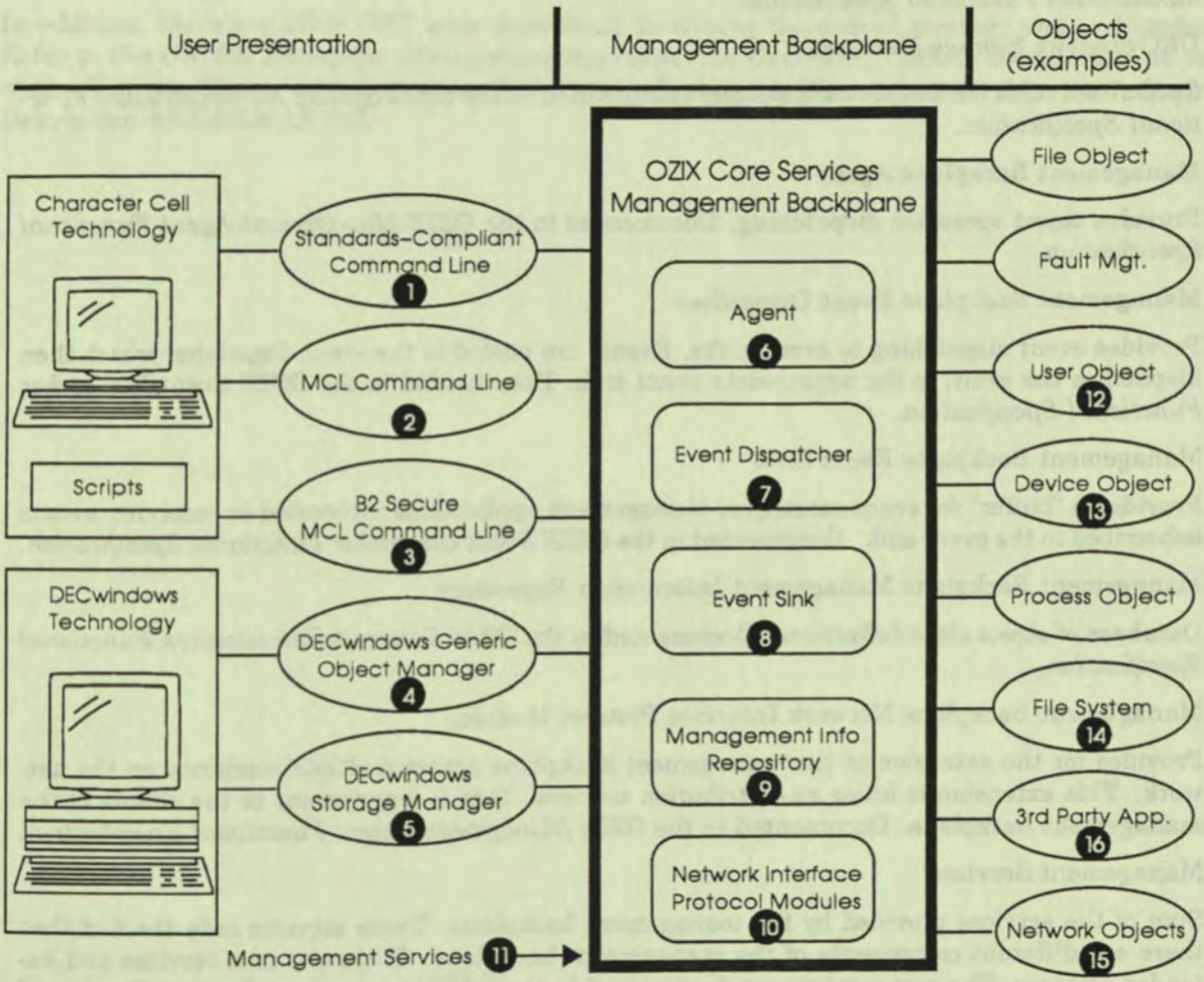
  Developers of these applications must choose how the applications fit into the management framework. As illustrated in Figure 9, third-party applications choose between connecting directly to the Enterprise Management Architecture (EMA) for a Digital specific solution or connecting with the management backplane through network protocols and standards. Each has its advantages and disadvantages.

  EMA provides added-value to the third-party application in that EMA provides better integration with Digital applications and a richer management application development tool set. The disadvantage is that the applications are specific to a Digital system. The converse of the above is true if the third party chooses to connect to the management structure using network protocols and standards.

3.6 OZIX System Administration Component Overview

This section provides the reader with a road map of OZIX system administration components and the specifications. The specifications describe details about each component. Figure 10 illustrates the components of system administration. The objects listed are only examples, unless otherwise stated.

Figure 10: System Administration Components



❶ Standards-Compliant Command Line

Command lines that are being provide to meet industry standards. Documented in the *OZIX System Administration Functional Specification*, *OZIX OSF Commands*, and *OZIX BSD Commands*.

❷ MCL Command Line

Generic command line to manage any manageable object. Documented in the *OZIX System Administration Functional Specification* and *Digital Network Architecture Network Control Language Specification, Version T1.0.0*.

❸ B2 Secure MCL Command Line

Generic command line to manage any manageable object in a B2 secure environment. Documented in the *OZIX System Administration Functional Specification* and *Digital Network Architecture Network Control Language Specification, Version T1.0.0.*

❹ DECwindows Object Interface

Generic object window manager DECwindows Interface. Documented in the *OZIX System Administration Functional Specification.*

❺ DECwindows Storage Manager

Specialized ABA DECwindows manager. Documented in the *OZIX System Administration Functional Specification.*

❻ Management Backplane Agent

Provides object operation dispatching. Documented in the *OZIX Management Agent Functional Specification.*

❼ Management Backplane Event Dispatcher

Provides event dispatching to event sinks. Events are posted to the event dispatcher which then dispatches the event to the appropriate event sink. Documented in the *OZIX Event Dispatcher Functional Specification.*

❽ Management Backplane Event Sink

Provides a "buffer" for event messages. Management applications interested in receiving events subscribed to the event sink. Documented in the *OZIX Event Dispatcher Functional Specification.*

❾ Management Backplane Management Information Repository

Database of object class definitions. Documented in the *OZIX System Administration Functional Specification.*

❿ Management Backplane Network Interface Protocol Modules

Provides for the extension of the management backplane across multiple machines on the network. This extension is know as *distribution services*. This is transparent to the clients of the management backplane. Documented in the *OZIX Management Agent Functional Specification.*

⓫ Management Services

Sum of the services provided by the management backplane. These services hide the fact that there are different components of the management backplane. There are core services and extended services. The core services are documented in the *OZIX Management Services Functional Specification.* Extended services are undefined in this version.

⓬ User Object

Represents the abstract of users and groups. Documented in the *OZIX System Administration Functional Specification.*

⓭ Device Object

Represents devices and containers. Documented in the specifications for the I/O subsystems which implement the objects.

⓮ File System

Represents the file system and data spaces.

⓯ Network Objects

Represents the manageable entities in the network. Documented in various network specifications. Refer to the Network Implementation Architecture chapter.

⓰ Third-Party Applications

Provides for applications that wish to connect to the management backplane. Documented in the *OZIX Guide to Writing Manageable Objects*.

In addition, there are other OSG wide documents describing the overall strategy and architecture. Refer to the *OS/SB Enterprise Management Approach: An Overview*. EMA is documented as part of the DECnet Phase V architecture. Refer to the *Enterprise Management Architecture: General Description #EK-DEMAR-GD*.

CHAPTER 4

RESOURCE/FAULT MANAGEMENT

4.1 Introduction

The major goal of OZIX Resource/Fault Management is to achieve OZIX product's availability, reliability, configuration and serviceability goals. OZIX Resource/Fault Management addresses:

- Error Detection Strategy
- Error Logging
- Fault Analysis
- Control the System Configuration

OZIX Resource/Fault Management concerns both hardware and software.

OZIX Resource/Fault Management is a single system strategy for error analysis and configuration; the strategy has been designed so that a higher level of management could be layered on top of the current single system strategy to manage a group of OZIX systems as a single system. OZIX support for management of multiple systems as a single system is not discussed in this paper, but may be addressed in a future version of OZIX.

4.2 Goals

OZIX product goals from "OZIX Product Features Document" for Resource/Fault Management are:

- Availability - One crash requiring operator intervention per year
- Configuration - Remove, detect, configure and bring-on-line devices
- Reliability - One Hardware/Software induced crash shutdown per year
- Serviceability - Maximize ability to repair and minimize time to repair

4.3 Overview

There are many components of OZIX resource and fault management.

OZIX resource management components are:

- Resource Manager
- User Interface

OZIX fault management components are:

- Error Monitor

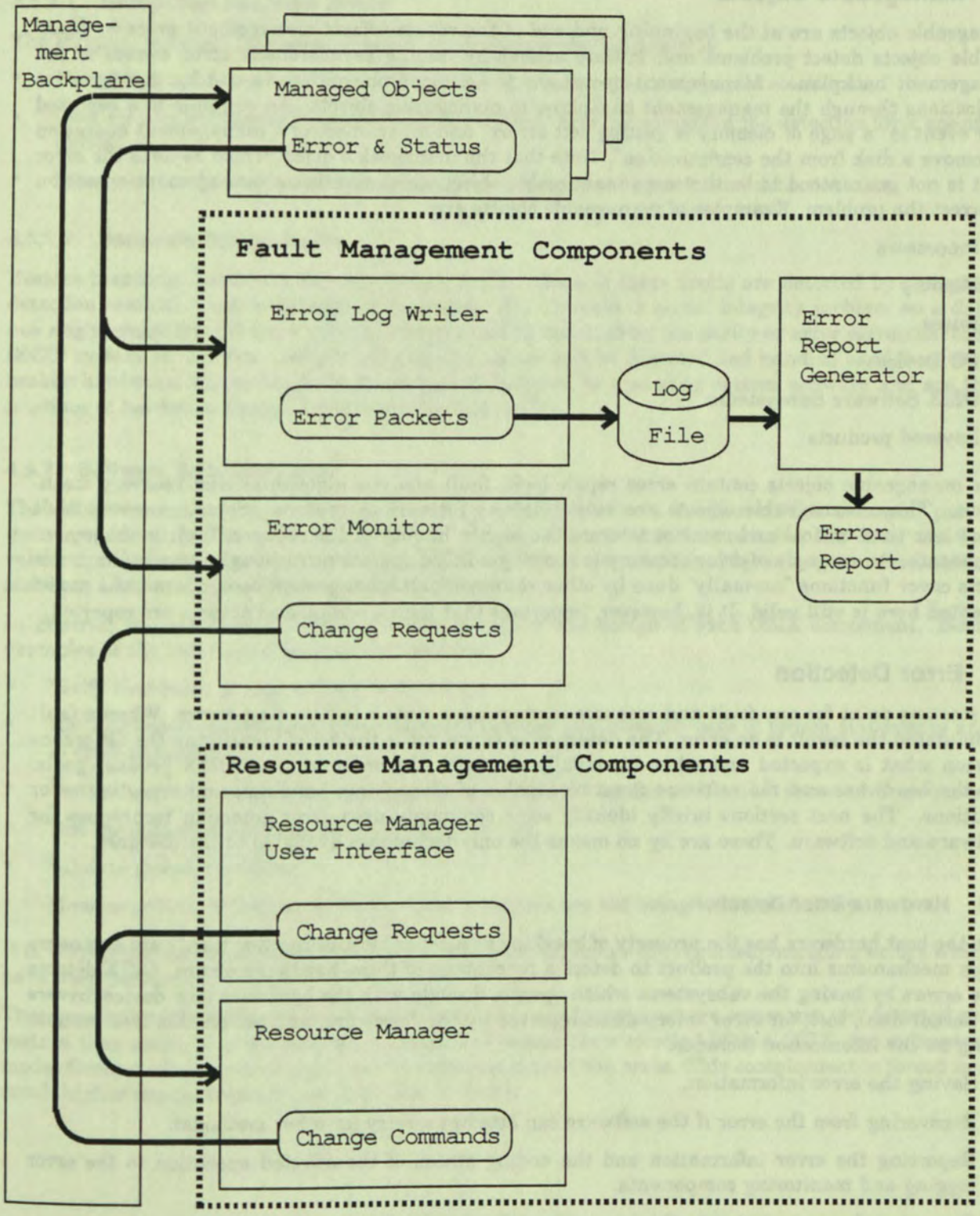
- Error Log Writer
- Error Report Generator
- User Interface
- Crash Dump Writer and Analyzer

Manageable objects work with both the resource and fault management components to achieve OZIX's goals.

The following diagram depicts the components involved in resource and fault management, how information flows between the components. OZIX System Administration Management Backplane is a secure channel over which the resource and fault management components communicate. The System Administration chapter describes the management backplane and its services. A simplified flow of information is:

1. **Error Information** is sent from the **Manageable Objects** to the **Error Log Writer** and the **Error Monitor**
2. The **Error Log Writer** writes the **Error Log File** on demand.
3. The **Error Report Generator** reads the **Error Log File** and writes the **Error Report**
4. **Change Requests** are sent from the **Error Monitor** and **User Interface** to the **Resource Manager**.
5. **Configuration Change Commands** are sent from the **Resource Manager** to the **Manageable Objects**

Figure 11: Resource/Fault Management Overview



4.4 Manageable Objects

Manageable objects are at the beginning and end of the resource/fault management process. Manageable objects detect problems and initiate actions by issuing asynchronous error events to the management backplane. Management operations to correct problems are issued by management applications through the management backplane to manageable objects. An example of a reported error event is "a page of memory is getting soft errors" and an example of a management operation is "remove a disk from the configuration". Note that the manageable object which reports the error event is not guaranteed to be the same manageable object which receives a management operation to correct the problem. Examples of manageable objects are:

- Processors
- Memory
- Buses
- I/O Devices
- OZIX Software Subsystems
- Layered products

Some manageable objects contain error repair logic, fault analysis algorithms and recovery mechanisms. These manageable objects are built this way for various reasons. Some error and fault actions are time-critical and can not tolerate the higher latency of the resource/fault management components. An example of error recovery is retrying a failed operation. Although these manageable objects cover functions "normally" done by other resource/fault management components, the model presented here is still valid. It is, however, important that faults, errors, and actions are reported.

4.5 Error Detection

The starting point for any fault and resource management system is detecting errors. When a fault is stimulated the result is an error. The detection of this error is the act of identifying the difference between what is expected and what is actually observed. In order to meet OZIX product goals, both the hardware and the software must be capable of identifying, handling, and reporting error conditions. The next sections briefly identify some commonly used error detection techniques for hardware and software. These are by no means the only techniques available to the designer.

4.5.1 Hardware Error Detection

Even the best hardware has the property of breaking at a definable failure rate. Hardware engineers design mechanisms into the product to detect a percentage of these hardware errors. OZIX detects these errors by having the subsystems which operate directly with the hardware (the device drivers and kernel nub), look for error information reported by the hardware, and act on this information. Acting on the information includes:

- Saving the error information.
- Recovering from the error if the software can attempt a retry (or other methods).
- Reporting the error information and the ending status of the affected operation to the error logging and monitoring components.
- Reporting the error status to the requester of the operation.

For the error monitor to work it is very important that all errors are reported, including errors which have been handled.

4.5.1.1 Unreported Hardware Errors

The strategy to detect hardware errors not reported by the hardware is:

- Use global techniques such has *timeouts* to detect some additional error conditions.
- Look for any additional detection mechanisms that may help improve the detection for a specific device.
- Select the hardware that OZIX supports based on its hardware failure detection rate.

4.5.1.2 Hardware Design Faults

Besides breaking, hardware also has design faults. Some of these faults are detected by the same detection methods that detect broken hardware. For example, a signal integrity problem on a data bus might cause one bit to be misread which could be detected by the parity or error correction code (ECC) system on the bus. Design faults of this nature will be detected and handled as if they were broken hardware. Some design faults cannot be detected by operating system software and are the province of hardware Design Verification Testing (DVT).

4.5.2 Software Error Detection

The detection methods for software are very similar to those in hardware. For example, parity memory adds an additional bit to each byte of data, defines rules for accessing the data, and the values that indicate an error. In software, error detection is accomplished by having some redundant states and knowing which states are error states.

In general, the detection of software errors is part of the design of each OZIX component. Some examples of the techniques that will be used are:

- Verify that routine arguments are reasonable.
- Have unknown parameters force execution through the "default" case of switch statements and to error handlers.
- Test for queue corruption by checking the type of data structure when it is removed from queue.
- Test for pool corruption by marking memory.
- Validate global variables.
- Monitor critical activities, so that critical resources are not being held indefinitely.

It is very important for the error monitor to work, that all errors are reported, including errors which have been handled.

There are a few "global" software detection mechanisms. In cases where errors are not detected the instant they occur, it is important to contain and isolate their effects. Within OZIX, the subsystem model forces containment of errors to the code that caused the error. This containment is forced in a much higher degree than in past operating systems.

4.6 Error Log Writer

The error log writer records error information. OZIX is using a generalization of the event dispatcher from Phase 5 of DECnet for dispatching error information. Since the needs of error logging are slightly different from network event logging, a few additional features are needed:

- Buffers for certain critical errors types must be locked in physical memory to be easily locatable in the crash dump. When OZIX crashes the error information must be recovered and written to the error log on the next reboot.
- Since the normal case for logging errors will be on the local system, some optimizations are needed to lower local error logging latency. These include treating the Local Sink and the Error Log Writer as special cases and:
  - Pre-configuring them for quick start up.
  - Bypassing DECnet.
  - Bypassing the conversions to and from ASN-1

This error/event dispatching mechanism provides a system that works well in the local case. It also allows the user to tailor a large complex of servers and workstations to use either centralized logging, local logging, or some combination based on device and error code. This also provide a port for any Field Service value-added service to monitor error information.

On each node the local Error Log Writer writes the error and event information to a local error log file. Source information includes the node so that a single file can contain data from many nodes.

4.7 Error Report Generator

The OZIX Error Report Generator is a support tool required for system fault analysis. It allows the system administrator or customer services engineer to obtain detailed information about events on the system and to create a report that is easy to read and interpret.

The Error Report Generator provides the user with a powerful troubleshooting tool by allowing the user to select and report related events. These events may be related by time range, event class (CPU, memory, network, system messages, etc.), device type, and logical or physical device name, as well as several other relations. It is possible for the user to correlate events that were flagged as related when they were written to the log. For example, if a device error causes both an MSCP error and an error at the file subsystem level, and these errors were flagged as related in the log, the Error Report Generator would write both of these entries when either was requested. (With all current Digital report generators, any such correlation must be made manually by the user.) Remote logging is an integral part of the error logging strategy and the log may contain event entries from many different nodes. It is possible to create reports of events on specific nodes.

The output format is also flexible. The user select entries to be written in several forms, from raw hex data to full bit-to-text translation of the entire record. The default output mode selects the most useful information from an entry and reports it in a concise form. A summary report is also available, providing counts of events over the specified range of time.

The Error Report Generator is a user application and reads the log file(s) written by the Error Log Writer. It selects specific entries from this file according to parameters supplied by the user. The selected entries are then formatted and a report written according to user specification.

The main goals of the Error Report Generator include:

- Accurate bit to text translations

- Intuitive and user friendly interface
- Modular design for the easy integration of additional event classes
- Good Performance - the user should not experience excessive delays
- Reliability - recovery and/or report of all errors incurred by the Error Report Generator such as error in input files, lack of resources, or user input.

The Error Report Generator is written in a highly modular fashion. This makes it possible to separate the bit-to-text translations of individual events from the rest of the code, allowing additional devices to be added easily in the future. Today, a major coding effort is required to add devices to the error log. In OZIX it should be possible to add additional devices and events by entering their entry definitions at a higher level and not by modifying and recompiling the source code.

4.8 Error Monitor

The Error Monitor has responsibility for fault analysis in an OZIX system. It is both a management application and a manageable object. As a management application it receives error and state change events from the management backplane and reacts by issuing management operations to the resource manager to correct an error or react to a state change.

As a manageable object it accepts management operations so that prediction algorithms, thresholds, and other decision making parameters can be modified within the error monitor.

The error monitor consists of a data base of errors and rule-based failure prediction algorithms. As errors are received, they are placed in the data base and the rules are tested to see if any conditions have been satisfied. Prediction algorithms in the rules can be simple thresholds like "a memory page is bad if 100 correctable read errors (CRD) memory occur in 1 hour in the page". Once a failure prediction is made, a request is sent to the resource manager to change the configuration to remove the component.

The error data base in the error monitor has knowledge only of manageable object sending errors. The database does not describe the configuration; the database that describes the configuration is maintained by the resource manager.

The rules in the error monitor are not in code but are maintained as a data file. A set of rules are supplied as part of the OZIX system. This allows addition and changes without requiring the error monitor to be recompiled. The error monitor rules handle devices at the physical level and not at logical or virtual levels.

Complex prediction algorithms can be built into the rules. These algorithms can correlate errors from different manageable objects to detect a fault which crosses manageable object boundaries. An example of such a correlation algorithm is determining a bus failure by looking at the errors of all the manageable objects connected to the bus. Multiple rules can be run on the same manageable object. For example, one rule could be thresholding on recovered errors on a given device and another rule could be thresholding on the total of all types of errors.

A rule can take different actions. The simplest action would be to tell the resource manager to remove a component from the configuration. Examples of complex actions would be to initiate diagnostics and exercise programs to attempt to stimulate and determine a fault. Other actions could include writing error information to stable storage on a board (EE PROM) and generating a service alert.

When the error monitor returns error information to the user interface, it does not return simple counts. Instead, the error values maintained by the rules may be displayed.

4.9 Resource Manager

The Resource Manager is responsible for configuration and recovery policy. It is both a management application and a manageable object. As a manageable object it receives management operations, some of which come from user presentations and others from the Error Monitor, to modify the configuration of the system. Additionally, the rules the Resource Manager uses to modify the system configuration are maintained through the manageable object interface the Resource Manager exports to the management backplane.

As a management applications it issues management operations to manageable objects, such as devices, in order to modify the configuration of the system. The decisions made by the Resource Manager as a management application are based on rules maintained by the Resource Manager.

The resource manager consists of a data base which describes the entire configuration, and rule-based recovery algorithms. As requests are received, the rules which apply to this request are tested. The management operations are sent to the manageable objects which are needed to implement the actions of the satisfied rules. The resource manager deals with devices at the physical level and not at the logical or virtual levels.

Resource manager rules are kept in a data file and not in code. A set of rules are supplied as part of the OZIX system. This allows addition and changes to be made to the rules without requiring the resource manager to be recompiled. Layered products such as Rdb™ Star, DECxTP can modify the rules to suit their environmental needs. Rules translate requests into management operations for the manageable objects. Resource manager rules can also make consistency checks, such as making sure the change will keep the system above the minimum configuration and checking for hardware and software version compatibility.

The resource manager database has information about all the elements in both the hardware and software configuration. Entries in the database contain information such as serial numbers and version numbers and other descriptive information. Also contained in the database is a history of recent changes. This allows analysis to be done between the current configuration and a previous configuration.

The resource manager replies to the request allows success or failure of fault recovery action to be returned. When the error monitor generates the service alert, it can then be determine if urgent service is required.

An important duty of the resource manager is version control. This is done by storing the revision information in the database and having rules which insure revision compatibility of components.

4.10 Recovery Mechanisms

Each OZIX component uses different recovery mechanisms. The OZIX system supplies common mechanisms, such as the transaction primitives. OZIX is also designed for recovery. For example, subsystem design encourages isolation and allows some subsystems which fail to be reloaded without the system going down. Each component can also implement its own mechanism and concept, including:

- Removing bad memory pages
- Disk shadowing
- ABA's movement of data from failing devices
- Fault recoverable based file systems

OZIX's design includes quick system restart and recovery.

Again it is very important that the manageable objects report actions taken.

4.11 User Interfaces for Resource/Fault Management

The Resource/Fault Management provides no user interface, per se. Instead, the manageable object interfaces exported by the Resource Manager and Error Monitor are presented to the human through user presentations provided by system administration. These interfaces include both a generic manageable object interface and a specialized user presentation for configuration of an OZIX system. For more information on the user presentations provided, see the System Administration chapter. Through these user presentations, the system administrator will not only be able to modify the configuration of the system, but also the rules used by the the Error Monitor and Resource Manager to configure the system.

In general, manageable objects, once taken off line by the resource manager, will only be placed back on line with intervention by the system administrator. This is done to track system configuration changes and the reasons for returning the manageable object to service.

The OZIX internationalization guidelines are followed in the design of the user interface.

4.12 Crash Dump Writer

When the OZIX operating system detects an internal inconsistency from which it can not recover, such as a corrupted data structure or an unexpected exception, it activates the system crash mechanism. This mechanism writes predetermined state information to dump file(s) and shuts down the operating system in a controlled fashion. This mechanism is also responsible for saving error information, in a previously allocated location. This information is sent to the event dispatcher to be recorded in the error log file upon system reboot.

The OZIX Crash Dump Writer does not rely on the failed operating system to perform any functions on its behalf. It is self contained and thus safe. To ensure that it has not been corrupted, the OZIX Crash Dump Writer code is checksummed prior to writing the crash.

The OZIX Crash Dump mechanism follows a hierarchical or multiple level approach to ensure that enough state information is saved. The system administrator has control over what level of dump is to be performed.

The OZIX Crash Dump mechanism writes the minimum amount of state information required to identify the failure to a small primary crash dump file that is located on the system disk. The primary file is not dilatable and is always present. All other selected levels will be written to a much larger secondary crash dump file. The secondary crash dump file size and location are variable and left up to the system administrator's discretion. The location can be any volume that is known to the OZIX operating system and is set by the system administrator. Management operations will be provided to manage the size and location of the secondary crash dump file.

A path to an alternate crash dump file is provided. This redundant path is used when the Crash Dump Writer is unable to access the primary path. This could be due to the device being off-line or inaccessible to the writer. Failover to the secondary path occurs transparently and does not require operator intervention. The OZIX Crash Dump Writer writes as much information as the file size will allow. When the Writer detects size limits it notifies the system administrator.

To avoid inconsistencies between the primary and the secondary crash dump files the Writer places a tag at the beginning of each file that is later matched by the analyzer. This will ensure that the files are synchronized. The Crash Dump Writer also places system image information, such as version number, into the crash dump file. The Crash Dump Analyzer then uses it for consistency checks with the image information it possesses.

4.13 Crash Dump Analyzer Description

The crash dump analyzer is a special version of the OZIX System Debugger with modifications that allow it to operate on the crash dump files and a running system memory in a read-only fashion. Refer to the detailed description of the OZIX System Debugger for further information.

The OZIX crash dump analyzer has an applicable subset of the system debugger's functionality along with crash analysis specific commands. Included in this is:

- At least ULTRIX MIPS™ dbx functionality
- Compiled language source display, when available
- Symbolic addresses
- Help
- History
- Script and log files
- Formatted data structures
- Language switchable (eg. Machine language translation)
- Format and display different stacks
- Display the locations and contents of procedure call frames
- Validation of the integrity of a queue by checking the pointers in the queue
- Consistency checks
- Display information about the state of the system at the time of the failure. This would include:
  - — Date and time of the crash
  - — Name and version of the O/S
- Display subsystem specific information
- Display executor specific information

The OZIX Crash Dump Analyzer operates in a client server relationship with the server portion residing in a specialized version of the system debugger nub. For communication the nub uses the OZIX Debug Protocol. Analysis is performed from either a remote node or locally from the rebooted system. This enables the OZIX Crash Dump Analyzer to have full operating support for reading source files, extracting symbols from system images, and displaying windows. For source level debugging all source files and system images must be accessible to the Crash Dump Analyzer. This method of crash dump analysis supports OZIX distributed environment.

CHAPTER 5

SECURITY

5.1 Scope

This chapter addresses the *security* and *integrity* issues involved in the design and development of OZIX. It discusses the security and integrity goals of OZIX, defines the elements of OZIX security and integrity , and describes the design features which allow the implementation of a secure OZIX operating system.

References are made throughout this chapter concerning *B2 Certification.* It is not expected that V 1.0 of OZIX will be either certified or certifiable. A later release of OZIX will be certified. However, the entire development and design structure to support certification will be in V 1.0. The elements missing for certification will be primarily documentation, security testing, and some support tools.

Throughout this chapter, new or unusual terms will be italicized on their first occurrence, and their definitions will be found in the glossary.

5.2 OZIX Security Goals

There are three broad sets of goals of the OZIX security design: basic security and integrity goals, extended security and integrity goals, and the global goals.

The basic security and integrity goals of OZIX include:

- Produce a system with significantly enhanced security, integrity and *robustness* relative to existing commercial systems.
- Be certifiable to the B2 level by the *National Computer Security Center (NCSC)*, as defined by Department of Defense (DoD) 5200.28.STD *(the Orange Book).*

The extended security and integrity goals of OZIX include:

- Support multiple different security and integrity models that are selectable by the customer. This will allow the *access controls* enforced by OZIX to model, as closely as possible, the customer's policies of accessing data.
- Allow the customer, at his discretion, to reduce the level of *enforcement* to enhance usability and performance.
- Provide the features required to permit applications running on OZIX to define and enforce their own access controls and to be incrementally certified.
- Be compatible with various models of distributed security.

The global security and integrity goals of OZIX include:

- Produce a high performance secure system

- Produce a highly usable secure system
- Produce a secure system whose security and integrity features are easily manageable

5.3 Basic Security and Integrity Goals

The security and integrity goals of a commercial operating system are:

- Preserve the *confidentiality* of sensitive information. Confidentiality is obtained by preventing sensitive information from reaching users who do not have a *clearance level* that permits them to handle such information. This type of protection is known as *Mandatory Access Control (MAC)* because it cannot be overridden by the owner of the information.
- Preserve the integrity of valuable information. Integrity is obtained by preventing valuable information from being altered or destroyed except by users at a *reliability level* that permits them to do so. This type of protection is known as *Integrity Access Control (IAC)* because it is designed to ensure the integrity of information.
- Allow the owner of data to restrict the distribution of his data, even from those with adequate clearance. This type of protection is known as *Discretionary Access Control (DAC)* because this control is applied at the discretion of the owner.
- Prevent any user or users from denying access to the system or its resources by authorized users. This type of protection is known as *Assurance of Resources (AOR)*.

Not only must the system satisfy these requirements in normal usage, but it must do so even when one or more *malicious* users are trying their worst to bypass the system protections.

MAC, DAC, and IAC can be thought of as three layers of access filtering that occur on any access attempt. The algorithms used by these layers to make their access decisions are the *security model*, *discretionary model*, and *integrity model*, respectively.

OZIX will be designed and developed to provide these features. In order to have an independent assessment of the security and integrity strengths of OZIX, and to provide our customers with assurances of the security and integrity features of OZIX, OZIX will be certified at the B2 level.

The B2 level of certification is considered appropriate because it provides a level of security and resistance to penetration that is adequate for all but sites involved in areas of national security.

The B2 level of certification is not considered excessive in terms of costs or other impacts, because it is not extremely difficult to validate and it imposes no limitations on export sales.

B2 Certification will assure that:

- A controlled development process and good software engineering techniques will be used. These factors are known to help create a more error-free system.
- A well known set of security related features as defined by the Orange Book will be implemented. These features will provide what is considered a minimum set which can assure system security.
- The strength of the system's security is quantifiable. A B2 system has a well defined strength, and can be compared to the strengths and assurances of other systems.
- The system's strength and resistance to penetration has been tested and proven by a board of expert independent evaluators.

These assurances are independent of the particular uses made of them. They can be used to implement a pure B2 system using the DoD model, or they can be used to implement a commercial system with the strength of a B2 system.

Most commercial customers would find a pure B2 system to be inappropriate for their needs. For this reason OZIX will implement an extended security and integrity model to meet the needs of our commercial customers, reserving the pure B2 model for those few applications where it is needed.

5.3.1 Implementing the Basic Protections

To state the requirements of a secure operating system more precisely: a secure operating system must control the access of *subjects* (things that perform actions) to *objects* (things upon which actions are performed) and *resources* (things used to perform actions).

This control is accomplished by:

- **Identifying** the fundamental subjects, objects, and resources
- **Isolating** all subjects from automatic access to any objects
- **Granting access** explicitly to appropriate objects
- **Controlling** usage of shared resources
- **Trusting** all code that implements isolation, access granting, or that deals with shared objects or resources
- **Recording** all accesses for later review to identify potential attacks on the system

The identification of trusted code leads to an additional access control constraint: data that is critical to maintaining the basic security and integrity of the system must be accessible only to code that is trusted enough to have that access. This access control is known *Trusted Access Control (TAC)*.

5.3.1.1 Identifying the Fundamental Entities

The fundamental entities that are relevant in OZIX are: subjects, objects and resources.

- Subjects include the following:
  - Executors—An *executor* is the OZIX generalization of a process encompassing all *threads* acting together in a single *execution context*. An executor has an identity and a clearance level that cannot be changed during its existence.
- Objects include the following:
  - Dataspaces—A *dataspace* is the OZIX conception of long-term persistent data on the system and is potentially accessible to any executor. Dataspaces are one of the primary security objects to which access must be controlled.
  - Sharable Memory Segments—Memory segments that can be mapped to the execution contexts of multiple executors are potential channels for data flow, and must have their access controlled.
  - Devices—Physical devices such as tapes, disks, printers and networks can act in some sense as a data repository. Access to these devices must, therefore, be controlled.
- Resources include the following:
  - CPU capacity–the compute power of the system
  - Memory space–physical memory pages
  - Disk space–file system, cache and paging space
  - I/O bandwidth–primarily the rate at which data can be moved to and from disk

- Network bandwidth–the rate at which data can be moved over the networks
- Physical devices–disks, tapes, terminals and network connections
- All other limited resources–anything shared and of fixed size or number

5.3.1.1.1 Labeling of Subjects and Objects

To implement access control, all subjects and objects must be labeled. These labels contain the identity of the subject or object, and all information required to make access decisions. These labels are located in memory areas not accessible to untrusted executors.

The elements that make up a label are:

- Unique Security Identity
- *Sensitivity Level* Range and *Compartments*
- *Integrity Level* Range and Compartments
- Ring Brackets

5.3.1.1.1.1 Unique Security Identity

Each security subject and object is assigned a unique identity. This identity is unique over the life of the system, and unambiguously identifies the subjects and objects in the audit log.

5.3.1.1.1.2 Sensitivity Levels

The sensitivity level of an object is usually used to determine what subjects can access an object, and in what manner. For example, a user cleared to the secret level would not be allowed to read top-secret data.

The sensitivity level is a hierarchical range, varying from highly sensitive (for example, top secret) to not sensitive (for example, unclassified).

OZIX will support 256 sensitivity levels. The human-readable labels associated with each level will be able to be set by the customer.

5.3.1.1.1.3 Sensitivity Compartments

Sensitivity compartments are usually used to control access to data on a "need-to-know" basis. For example, a user cleared to access security data would not necessarily be allowed to access payroll data.

Sensitivity compartments are a non-hierarchical set of classes of information. An object may belong to one or more of these classes. OZIX will support 1024 sensitivity compartments. The human-readable label associated with each compartment will be able to be set by the customer.

5.3.1.1.1.4 Integrity Levels

The Integrity level of an object is usually used to determine what subjects can access an object, and in what manner. For example, a user reliable at the malicious level would not be allowed to modify system-critical data.

The integrity level is a hierarchical range varying from highly reliable to malicious for users and from precious to junk for data.

OZIX will support 256 levels of integrity. The human-readable label associated with each integrity level will be able to be set by the customer.

5.3.1.1.1.5 Integrity Compartments

Integrity compartments are usually used to control access to data on a "need-to-modify" basis. For example, a user cleared to modify security data would not necessarily be allowed to modify payroll data.

Integrity compartments are a non-hierarchical set of classes of information. OZIX will support 1024 sensitivity compartments. The human-readable label associated with each compartment will be able to be set by the customer.

5.3.1.1.1.6 Security Rings

Security rings are used to limit access to system critical data to code that is trusted to access the data.

Security rings are a set of heirarchical levels ranging from 0 (completely trusted) to 255 (completely untrusted). All security objects in the system are labeled with two ring numbers: a read ring and a write ring. Any executor whose current ring execution level is at or below the data's read ring level is permitted to read (or execute) the data. Any executor whose current ring execution level is at or below the data's write ring level is permitted to write the data.

An executor's current ring execution level is determined by the *ring brackets* (read ring level and write ring level) of the code the executor is executing. Executing code may set its executor's current ring execution level to any value equal to or greater than that code's write ring level.

For example, if an executor is executing at a current ring execution level of, say, 50, then that executor could only execute a *Subsystem Procedure Call (SPC)* to code whose read ring level was 50 or higher. If a call were made to code whose ring brackets were [50,20], that code could set the executors ring execution level as low as 20. Upon return from the called code, the executor's ring execution level would be returned to 50.

Figure 12 illustrates how ring levels apply to OZIX *subsystems*, *packages* and data. For the sake of simplicity, this example will ignore the other access controls on the packages (MAC, DAC and IAC).

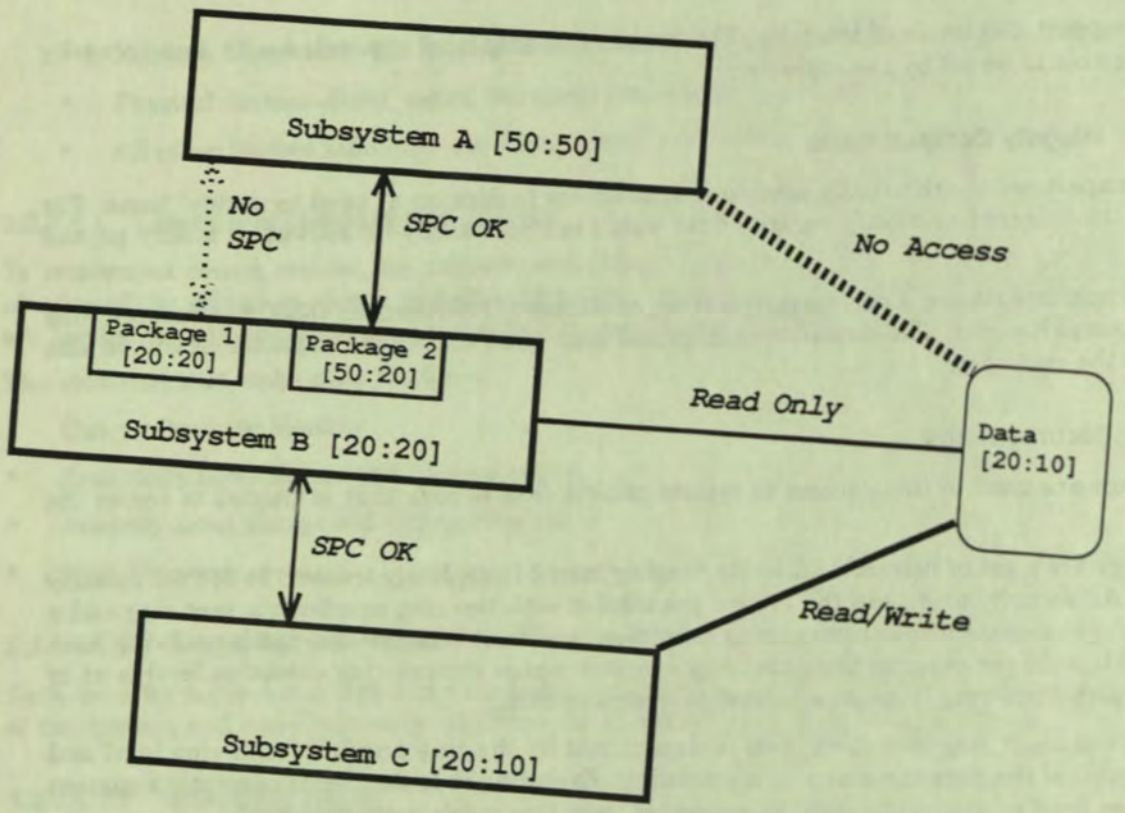
Three subsystems are illustrated: A, B, and C. Subsystem A has ring brackets of [50:50]. Subsystem B implements two packages: package 1, whose ring brackets are [20:20]; and package 2, whose ring brackets are [50:20]. The body of the subsystem B code is at ring brackets [20:20]. Subsystem C has ring brackets of [20:10]. Also shown in Figure 12 is a data object with ring brackets [20:10].

An executor in subsystem A is running relatively untrusted code at ring level 50. No executor in this subsystem can access the data object. An executor in subsystem A may not call subsystem C directly at all. Package 1 of subsystem B also cannot be called, though package 2 can be called.

Once an executor has entered subsystem B, the code in subsystem B can lower the ring execution level to as low as 20. At ring level 20, the data object can be read, but cannot be written.

Once at ring execution level 20, a call can be made to subsystem C. Subsystem C can lower the ring execution level to 10. At level 10 the data object can be both written and read.

Figure 12: Security Ring Levels



5.3.1.2 Ensuring Isolation

The actual isolation in OZIX is provided by the hardware protection of memory. All other isolation mechanisms are built on top of this memory protection. The software isolation mechanisms are designed such that any executor attempting to violate the security or integrity models will violate the hardware memory protections and be vectored to the *Nub*.

Executors are isolated from each other's data because each executor executes in a separate execution context. (Loosely, an execution context can be thought of as an address space). It is impossible for an executor to reference any memory that is not in its execution context. Any address that an executor can state is within his own virtual address range. If no physical memory is at the specified address, or if the access mode is not appropriate for the specified address, the hardware protection mechanisms will trap the access attempt.

This isolation by execution context extends to much more than the virtual addresses that can be seen by application level code. Many of the subsystems that implement OZIX use virtual memory segments that are selected depending on the particular executor running in the subsystem. Thus, even these subsystems cannot reference addresses that do not belong to the running executor.

5.3.1.3 Granting Access to Objects

The *access validation component* is the mechanism whereby an executor is granted access to additional objects. A request, for example, to add a shared segment to an executor's execution context will result in a call to the access validation component. The access validation component will apply each of the defined access filters (security model, discretionary model, integrity model, trustedness model) to the request, and, if the access passes all the filters, permit the mapping to occur.

5.3.1.4 Controlling Usage of Resources

The general goal of controlled resource usage is to prevent a user or group of users from denying service to other users. This statement of the goal, however, seems too simple for a system designed to run in a wide variety of environments. In some circumstances it may be acceptable (indeed, required) that a highly critical task be performed, even if it consumes the entire system. In the extreme cases, it may even be acceptable if other, less critical tasks are aborted.

Thus OZIX implements a new executor attribute, *priority*, and restates the goal as: "No user is permitted to deny a user of higher priority access to system resources".

In order to implement this goal, all usage of all limited system resources will be controlled. Some such resources are obvious: CPU cycles, memory, disk space. Others are not so obvious: bandwidths, fixed size system structures, locks.

Any subsystem that controls a shared resource will prevent denial of resources attacks. This implies that usage records, quotas and a priority algorithm will be enforced for any such shared resources.

5.3.1.5 Trusting Code

Code must be trusted either because it must be correct or because it must be discreet:

- Correct code implements the pervasive security features that control isolation and access—such as the access checking mechanisms —such code must be correct or the pervasive security features would not work
- Discreet code, by the nature of its function, must access data from multiple levels or users—such as the dataspace cache that caches data for all users and levels—such code must be discreet since it **could** disclose information.

Such portions of the system must be trusted to perform their functions correctly and not disclose data inappropriately. Trust is not earned easily. In a certified secure operating system trusted code must be developed using good design practices, and its design and code must be carefully reviewed. Trusted code must be subjected to extensive testing and documentation.

Any portion of the system that is trusted becomes a member of the *trusted computing base (TCB)*. Since considerable effort is involved in trusting a piece of code, a goal of OZIX is to minimize the amount of trusted code and, hence, the size of the TCB.

5.3.1.6 Recording Access Events

Various types of security relevant events must be able to be audited by the TCB. The intended function of the audit facility is to permit the security officer to detect and analyze break-in attempts.

The types of events that will be audited include:

- Use of *authentication* mechanisms
- Addition or deletion of objects from a user's address space

- Actions taken by the operator or administrators
- Production of printed output
- Override of human readable output markings
- Change of designation of any communication channel or I/O device
- Events that may exercise *covert storage channels*

Tools to analyze the recorded audit information will be developed.

5.3.2 Obtaining Certification

Obtaining a certification from the NCSC requires:

- Developing the system in accordance with the NCSC's guidelines *(the rainbow books)*
- Designing the system to include the NCSC required security features
- Producing security related documentation
- Verifying that the system meets the NCSC guidelines
- Performing security related testing
- Submitting the system to the NCSC for approval
- Passing NCSC penetration testing

5.3.2.1 Development

At the higher levels of certification, security and integrity must be designed into a system from the start. Add-on security may be acceptable for lower levels of assurance, but this is not true at the B2 level and above. Indeed, the very structure of a B2 system is considered part of the security assurances.

5.3.2.1.1 Methodology

Development of a certified system must follow "good" software engineering practices. This has been generally interpreted to mean using the specification, design, code model with appropriate reviews, inspections and testing.

These software engineering technologies are accepted as providing a system which with fewer errors, more consistent implementation, and better maintainability than systems developed using ad hoc methods.

The development methodologies used on OZIX are documented in the *OZIX Software Development Procedures* listed in [5] in Section 5.7.

5.3.2.1.2 Configuration Management

The documents and code related to a certified system are expected to be under configuration control during the entire life cycle. This is intended to allow traceability of changes (both design and code), and to prevent trojan horses, trap doors, etc. from being inserted into the system.

5.3.2.2 Required Security Features

The Orange Book requires:

- Identification, analysis, control, test and audit of covert channels
- Formal and proven security policy model
- Good development practices including configuration management
- Detailed top level specification (DTLS) for the TCB
- Support for *multi-level devices*
- Mandatory access control for all subjects and objects
- Separate domain of execution for the TCB
- Modularized TCB
- Principle of *least privilege*
- Identification and documentation of all internal and external TCB interfaces
- Secure TCB generation
- Trusted path for login and some other security functions
- Auditing of security relevant events

5.3.2.3 Documentation

The following documents must be produced for evaluation by the NCSC:

- Security Features User Guide–A user document that describes the protection mechanisms provided, guidelines on their use, and how they interact with one another.
- Trusted Facility Manual–A system administrator's manual that describes the administrator and operator actions related to security, guidelines on consistent and effective use of the protection features of the system, and procedures for secure generation of a new TCB from source after modification.
- Test Documentation–A document that describes the test plan, test procedures and results of the testing of the security mechanisms. It must include results of testing the effectiveness of the methods used to reduce covert channel bandwidths.
- Design Documentation.

  The following are required:

  - The TCB interface description.
  - A formal security policy model description.
  - A descriptive top-level specification–A complete description of the TCB including exceptions, error messages, and effects.
  - Miscellaneous documentation.
    - Description of the reference monitor implementation and an explanation of why it is tamper-resistant, cannot be bypassed, and is correctly implemented.
    - Description of how the TCB is structured to facilitate testing.
    - Description of how the TCB is structured to enforce least privilege.

- Results of covert channel analysis and the tradeoffs involved in restricting the channels.

5.3.2.4 Verification

The following techniques must be used to verify the specified characteristics of the system:

- Analyze–the design documentation, source code, and object code.
- Prove–that the formal model of the *security policy* is consistent with its axioms.
- Show–that the descriptive top-level specification of the TCB completely and accurately describes the TCB and provides an accurate description of the TCB interface.

5.3.2.5 Testing

The following security-specific testing is required:

- The security mechanisms must be tested and found to work as claimed in the system documentation.
- Testing must demonstrate that the TCB implementation is consistent with the descriptive top-level specification.
- The effectiveness of the methods used to reduce covert channel bandwidths must be tested and prove to function correctly.

5.3.2.6 NCSC Approval

The overall system development, design, testing and documentation must be approved by the NCSC in order to obtain certification. The NCSC *Vendor Assistance Program (VAP)* provides a review team of evaluators during early system design phases to help a vendor produce a certifiable system by answering questions and reviewing designs and documents.

5.3.2.6.1 NCSC Penetration Testing

Armed with all system documentation (including code), the NCSC will assign a team to attempt to bypass the security features of the system. The penetration team will usually consist of about three experienced evaluators.

Digital will be required to supply a test system for the use of the evaluators, either at our site or theirs.

5.4 Extended Security and Integrity

Meeting the basic security and integrity goals is not enough. A system that met only these goals would indeed be a B2 secure system, but it would not satisfy the real requirements of our customers and application developers. OZIX supplies a set of features that permit a customer or application developer to tailor and extend the security and integrity characteristics of the system to fit the customer's needs.

5.4.1 Alternate Models

For purposes of certification, OZIX will be equipped with a slightly modified version of the Bell and LaPadula Security Model. This model satisfies the requirements of DoD security and its mathematical properties are well known within the security community. This model is not a desirable model for the majority of commercial customers because it excessively limits the capability to share data.

OZIX will permit a customer to install alternative access control models to replace the standard models. These models can be tailored to fit the way that the customer views confidentiality and integrity of data.

Section 5.6.2 describes an alternate set of access controls that might be used by a commercial customer. This simplified model is patterned on the way that Digital handles confidential data.

5.4.2 Alternate Authenticators

OZIX consolidates all authentication in one place, the Access Validation Subsystem, both for added security, and for ease of system management. Since this scheme prevents a customer from simply replacing or adding programs to perform custom authentication, OZIX provides the capability to add additional authentication mechanisms to the Access Validation Subsystem to meet his sites requirements. These mechanisms can be used at login, or for identity authentication from the application level.

For example, if a customer wishes to use *smart card* authentication, a new authenticator which supported this type of identity validation could be added to the access control subsystem.

5.4.3 Enforcement Levels

There are a number of security checks and controls that taken together enforce system security and integrity. OZIX provides the capability to increase performance (at the cost of reduced security and integrity) by selectively disabling these controls. The following are the security and integrity enforcement switches that the customer can disable at his discretion.

- Covert Channel Delays–A value that controls the amount of delay used to limit covert channel bandwidth
- TCB Stacks–Enable creation of a new stack when entering the TCB
- Auditing Control–Audit selections:
  - Levels–The level of significance an event must have to be audited
  - Events–Particular event types that are to be audited
  - Requested–Determines whether to honor ACL requested audit events
- Isolate Address Spaces–Enables isolation of address spaces in separate subsystems
- Device Sanitization–Enables complete reinitialization of devices and terminals when changing labels
- Integrity Access Control–Enables checking of IAC labels upon access
- Exact Quotas–Use *mutexes* when updating/checking quotas
- Quota Checking–Quotas limits for individual resources may be separately selected or disabled
- ID Stack–Enables id stack updates during gate crossings to allow context sensitive IAC
- Discretionary Access–Enables DAC checking using access control lists

- Mandatory Access–Enables MAC label checking upon access
- Trustness Access–Enables TAC label checking upon access

5.4.4 Features to Support Secure Applications

A goal of OZIX is to supply the features required to allow secure applications to be implemented on top of the operating system. Ideally, these applications could be incrementally certified to any level at or below B2.

OZIX implements four features to aid in this goal: *security classes*, *access stacks*, *ring levels*, and *identity stacks*.

5.4.4.1 Security Classes

OZIX permits the creation of security classes. A security class is a set of objects whose accesses are controlled by a set of MAC, DAC, IAC, TAC and other models. It is possible in OZIX to define a new security class whose accesses are controlled by a discrete set of access filters.

Subjects belong to a set of classes and may access objects only in the classes to which they belong.

The capability of defining new security classes permits applications to create new and different types of access controls which will be enforced with the full strength of OZIX's B2 security.

For example, a database application might choose to implement a new class of objects called "database objects". These objects could co-exist with normal objects, while remaining inaccessible to non-database users. The access controls enforced on the database objects might be entirely different from the access controls enforced on the normal objects.

5.4.4.2 Access Stacks

As mentioned previously (Section 5.3), the access controls applied to a security class can be considered as a stack of access filters. OZIX permits the addition of new filters into the access filter stack of any security class.

This capability permits the creation of new and different access controls which are enforced with the full strength of OZIX's B2 security.

Some examples of uses of this feature would include:

- License Access Control—accesses can be permitted or denied depending on whether the access is permitted based on the installed licenses.
- Timed Access Control—access to certain objects may only be permitted at particular times of day, or on certain days
- Extended Auditing—a null access filter (one which always permits access) can be installed which performs additional or special auditing, such as across the network to a write only auditor

5.4.4.3 Ring Brackets

The ring bracket mechanism can be used by applications to implement their own *security kernel*. Just as the ring brackets allow OZIX to protect the files and structures which are required for its correct operation, applications can use ring brackets to protect their own files and structures from everyone except OZIX—and OZIX can be trusted not to corrupt them.

This capability allows, for example, a database application to protect its own user password file with B2 level assurance that users cannot corrupt it.

5.4.4.4 Identity Stack

OZIX supports the concept of an identity stack. This stack can be considered to be a list whose first element is the user's name, and each subsequent element is a component identity. At any instant, the identity stack shows the entire call history of the current execution. For example, if user JIM, running a Korn shell, calls a database application which in turn calls the API subsystem which calls the file name subsystem, during execution in the file name subsystem, the identity stack would be:

JIM.KSH.RDB_STAR.OSF_API.FILE_NAME_SS

DAC checks for object access can perform pattern matching operations on the identity stack. For example, an ACL might permit access to a file only if the user identity were "JIM", and the identity "RDB_STAR" were immediately preceding OSF_API on the identity stack.

This capability allows applications to limit access to objects to only the code which is expected to manipulate them.

5.4.5 Distributed Security

OZIX will not initially support network distributed security. However, the mechanisms for adding this support are available. Alternate authenticators can be easily added to the access validation subsystem and used by applications and system services for distributed authentication.

5.5 Global Goals

Security and integrity enforcement cannot be free of performance impact. Additional computation is required in performing access checks, access to new databases must be performed, and inefficiencies may be introduced by the structuring of the system The challenge is to minimize the visible impacts of security and integrity enforcement.

The design philosophy of OZIX is to "minimize the mainline". That is, not to add security relevant code to the normal execution paths. OZIX will rely heavily on cached access checking. Once a user has been given access to something, that access need not be checked again. OZIX will be designed to contain the majority of the access checking impacts to the initial access attempt, and to have a minimal impact for the duration of processing.

Security and integrity enforcement cannot be free of usability impact. Additional constraints and limitations are imposed, and additional features are supplied which must be managed. The challenge is to minimize the visible impacts of security and integrity enforcement.

The design philosophy of OZIX is to "minimize the additional actions required". That is, not to add security relevant actions to normal execution paths. OZIX will rely heavily on defaults to provide security and integrity information. Once a user has set up his security and integrity configuration, little additional will be required of him.

Data sharing cannot be as unconstrained in a secure system as in an insecure system. The very nature of security and integrity enforcement is to add "cannots" to a system. This is not something that can be designed around. OZIX will minimize the impact of reduced data sharing by allowing the customer to specify what the sharing model is to be, and not force an overly restrictive model on the users.

Security and integrity enforcement can not be free of system management impact. Additional features, attributes and controls are added as a result of security and integrity enforcement, and these must be managed. The challenge is to minimize the visible impacts of security and integrity enforcement.

OZIX will minimize the system management impacts of security and integrity in several ways. OZIX will use a consistent management approach across all aspects of the system, including enforcement. Thus a system administrator will not have to learn to use tools which are unique to managing the system enforcement. OZIX will integrate many of the security and integrity controls into common objects with common interfaces in place of the ad hoc databases used in many systems. This will present a unified interface to the system administrator at a single point, instead of multiple types of interfaces applied to many different features. Finally, OZIX will depend on common standard default enforcement configurations and models, which will permit a customer to install a consistent system with very little tuning or modification.

5.6 Examples of Security and Integrity Models

The following sections will compare and contrast two different security and integrity models with which OZIX could be equipped. The first is a certifiable Department of Defense (DoD) model, the second a commercial security model.

5.6.1 Department of Defense Security Model

This section describes a security model usable by the Department of Defense (DoD) and which will be the basis of the certified OZIX product.

Table 2 defines the security and integrity labels which are used by this access control model.

Table 1: DoD Model Labels

Levels	Compartments
Security Levels	Security Compartments
UNCLASSIFIED	NATO
CONFIDENTIAL	CRYPTO
SECRET	NUCLEAR
TOP_SECRET	etc. etc.
Integrity Levels	Integrity Compartments
none	none

OZIX will be certified using a modified Bell and LaPadula model for mandatory access control using these labels. This model states:

- **A user cannot read data that is at a higher sensitivity level than the user's own sensitivity level.**

  For example, a user classified at the secret level cannot read top secret data. This restriction is known as the *simple security policy*. It prevents a user from reading data he is not cleared to see.

- **A user cannot write data that is not at the user's sensitivity level.**

  For example, a user classified at the secret level cannot write data into an unclassified file. This restriction is known as the **-property*. It prevents a user from accidentally or intentionally placing data where it could be read by someone not cleared to see it.

- **A user cannot read data unless he is cleared to access all of the compartments of the data.**

  For example, a user classified at the secret level and belonging to the compartments NATO and CRYPTO cannot read secret data belonging to the NATO and NUCLEAR compartments. This restriction is known as *need to know*. It prevents a user from seeing data that is not required to perform his assignment.

- **A user cannot write data that is not in all of the user's compartments.**

  For example, a user belonging only to the compartments NATO and CRYPTO cannot write a file belonging only to the NATO compartment. This restriction prevents a user from accidentally or intentionally placing data where it could be read by someone without the "need to know" that data.

The Bell and LaPadula model contains no rules concerning integrity. In general, the models used for NCSC certification use no integrity models.

5.6.2 Commercial Access Control Example

This section describes a simple access control model which would be applicable in many commercial development environments, that can be implemented using the extended security and integrity features of OZIX.

Table 2 defines the security and integrity labels that will be used by this access control model.

Table 2: Commercial Model Labels

Levels	Compartments
Security Levels	Security Compartments
UNCLASSIFIED	GENERAL
INTERNAL_USE_ONLY	SYSTEM
CONFIDENTIAL_and_PROPRIETARY	FINANCE
	PREPARED_by_ATTORNEY
Integrity Levels	Integrity Compartments
UNKNOWN	GENERAL
LOCAL	PRODUCTION
VITAL	TEST
	DEVELOPMENT
	SYSTEM

The security levels represent the levels of sensitivity common in a commercial corporation.

The Security compartments represent classes of data that may be accessible on a "need-to-know" basis.

The integrity levels represent levels of trust of code, and value of data.

- VITAL code is code that is highly trusted. This usually consists of stable, commercially produced code that is installed by the system administrator. VITAL data is data of high value, whose loss would have significant impact, such as system configuration databases or major work products.

- LOCAL code is code that is somewhat trusted. It is usually produced by normal system users and that is believed to operate correctly. LOCAL data is data of intermediate value, whose loss would have some impact, such as memos or executables.
- UNKNOWN code is code that is potentially malicious or incorrect. This category of code would include code obtained from random outside sources, and known "buggy" code. UNKNOWN data is data of little or no value whose loss would have little impact, such as temporary files.

The Integrity compartments represent classes of data that may be accessible on a "need-to-modify" basis.

Access between subjects and objects are based on these levels and compartments. The rules that define allowable accesses are the security and integrity models.

The security model rules are:

- **A user cannot read data that is at a higher security level than the user's own.**

  For example, a user cleared at the INTERNAL_USE_ONLY level cannot read CONFIDENTIAL_and_PROPRIETARY data. This restriction prevents a user from reading data he is not authorized to see.

- **A user cannot read data unless the user is authorized to access all of the compartments of the data.**

  For example, a user belonging to the compartments FINANCE and PREPARED_by_ATTORNEY cannot read data belonging to the FINANCE and SYSTEM compartments. This restriction prevents a user from seeing data that is not required for his assignment.

- **A user cannot write data that is not in all of the user's compartments.**

  For example, a user belonging to the compartments FINANCE and GENERAL cannot write a file belonging only to the GENERAL compartment. This restriction prevents a user from accidentally or intentionally placing data where it could be read by someone without the "need to know" that data.

- **Low integrity code may not read high sensitivity data.**

  For example, a program of UNKNOWN integrity (imported from a billboard, perhaps) may not read CONFIDENTIAL_and_PROPRIATARY data. This restriction prevents *Trojan Horse* programs from reading and re-distributing sensitive data.

The following are the integrity model rules:

- **A user cannot read data that is at a lower value level than the user's own reliability level.**

  For example, a user at the LOCAL reliability level cannot execute (read) an UNKNOWN reliability program. This restriction prevents a program from an unknown source from acting as a Trojan Horse and destroying valuable data.

- **A user cannot write data that is at a higher integrity level than the user's own reliability level.**

  For example, a user classified at the LOCAL integrity level cannot write data into a file classified at VITAL integrity. This restriction prevents a user of limited integrity from contaminating data that must be highly trusted to be correct.

- **A user cannot write data unless the user is authorized to access all of the compartments of the data.**

For example, a user belonging to the compartment TEST cannot write data belonging to the PRODUCTION and TEST compartments. This restriction prevents a user, who is performing testing, from modifying production data with possibly incorrect code.

Table 3 compares the modified Bell and LaPadula access control model described in Section 5.6.1 and the simple commercial access control model defined here.

Table 3: Comparison of DoD and simple Commercial Access Control Models

DoD Model	Simple Commercial Model
Security Model (Modified Bell & LaPadula)	**Security Model**
Subject may not read higher level object	Subject may not read higher level object
Subject may not write any other level object	
Subject may not read objects in other compartments	Subject may not read objects in other compartments
Subject may not write object with fewer compartments	
Only security officer may downgrade	Object owner may downgrade
	Low integrity code may not read high security data
Integrity Model	**Integrity Model (Simple Biba)**
None	Subject may not read lower level object
	Subject may not write higher level object
	Subject may not write objects in other compartments

The view of security in a commercial environment is quite different that that in a DoD environment. In a DoD environment, users are not generally trusted not to disclose sensitive data. The whole DoD security structure is designed to prevent sensitive data disclosure, even by a malicious user.

In a commercial environment, however, the view of security may be quite different. In a commercial development environment, such as the Digital environment for example, the users are generally trusted not to disclose sensitive data. The primary worry is the theft or destruction of data by a malicious penetration. Other types of commercial environments will have different security requirements.

The differences between the above two access control models clearly the more relaxed security controls of a commercial development environment. Two major access restrictions have been removed from the DoD model in creating the commercial model. Data sharing and ease of use are considerably enhanced using this model as compared to the more restrictive DoD model.

The addition of an integrity model for access control reflects the fact that data is a valuable resource to a company, and must be protected from destruction or corruption, as well as being protected from disclosure.

5.7 Related Reading

[1] *Department of Defense Trusted Computer System Evaluation Criteria*,–, DOD 5200.28.STD, December 1985.

[2] *Building a Secure Computer System*, Gasser, Van Nostrand Reinhold Co., 1988.

[3] *Distributed System Security Architecture Requirements*, Gasser, Goldstein, Kaufman, Lampson, Digital, Mar 1989.

[4] *Preliminary ABA Architecture Overview [Bean], rev 0.5*

[5] *OZIX Software Development Procedures*, Digital, Aug 1989.

CHAPTER 6

INTERNATIONALIZATION

6.1 Introduction

This chapter explains OZIX goals for internationalization and the design model used for OZIX software, and includes descriptions of individual OZIX international components. The purpose of this chapter is to provide a frame of reference from which to examine the design documentation of individual OZIX components.

6.1.1 OZIX Description

OZIX is an international product that provides a superior implementation of open systems software. The OZIX design takes into account requirements for compliance to open systems standards as well as requirements for localization and international capabilities.

All OZIX components facilitate translation by structuring program user interfaces and message text separately from functional code and by using multilingual messaging and language switching facilities. The OZIX base system software uses compound string technology to support multilingual software, and is code-set independent to the level of terminal services. Compound string technology is also incorporated into the OZIX file system, libraries and programming tools.

6.1.2 Terminology Definition

A definition of terminology is appropriate to begin this chapter. *Internationalization* is the process of developing an international product and delivering that product into worldwide markets. Internationalization incorporates two concepts—*localization* and *international capabilities*.

Localization is the process of adapting an international product to suit the language, conventions, and market requirements of a particular *locale* or local environment. Locale-specific conventions include cultural data representations such as radix and currency symbols, and formats for date, time, and calendar. Also dependent on locale are conventions for collating and sorting, keyboard mappings, character sets and character code sets, and conversion functions. *Translation*, which is the rendering of information presented in one natural language into another, is part of localization. Localization does not change the functionality of the international product.

International capabilities are those functions of an international product that support the languages and conventions of more than one locale. International products may be *localizable* or *multilingual*. Localizable software is software that can be modified to suit particular locales, while multilingual software has one version of software supporting multiple locales at the same time.

6.2 Goals

In concert with Digital's goal of providing worldwide software products, OZIX V1.0 is engineered with clear and pervasive internationalization goals. These goals may be classified as either design goals or functional goals. The functional goals are the following:

- Support for easily localizable applications
- Support for multilingual applications
- Good run-time performance
- Object level code compatibility with existing ULTRIX (RISC) applications
- Language and culture neutral base system components

The following are design goals in OZIX:

- Straightforward requirements for packaging, installation, and system management with respect to localized products
- Limiting locale-specific processing of internationalized data to the highest layer of software possible, typically at the presentation and user interface layers
- Logical decoupling of components such that localization can take place on a component by component basis to suit the local market requirements

The OZIX base system itself does not need to be multilingual, but it is important that the base system not preclude support for the execution of multilingual applications. In addition, not all base components must be localizable. For example, it may not be productive or possible to localize the console output produced during system crash recovery.

6.3 Design Model

The OZIX design for international support is a superset of several prior efforts to solve the general problem of creating international software. However, the requirement for OZIX to be an open system strictly tempers its design for internationalization, because OZIX must provide standard programming interfaces for traditional UNIX applications and interoperate with existing UNIX systems. The following sections explore the problems for which OZIX supplies solutions, and present the elements of OZIX internationalization with respect to previous and current design efforts.

6.3.1 Problem Statement

While American and European language versions of new software products are generally delivered to the local markets in a timely fashion, current engineering practices and projects do not produce products that can be easily localized for Far Eastern and Middle Eastern markets. Instead, additional engineering is required to build a product suitable for localization. This reengineering effort may include redesign of the product to handle additional character sets and screen display requirements. Reengineering, normally performed by local engineering groups in Japan, Hong Kong, Israel, and by International Engineering Development (IED) in Reading, adds to the cost of the localized product and seriously delays introduction of the localized product into the marketplace.

Digital is aggressively working to solve the problem of producing not only translated components, but also multilingual systems. For instance, *DEC STD 066-3* identifies the countries currently considered to be Digital strategic markets, and it is a corporate goal that new products be capable of simultaneous shipment to these strategic markets.

Digital is not alone in its effort to define solutions for internationalization problems. Various standards bodies are also attacking the problem. The American National Standards Committee X3J11 standard for the C Programming Language (Draft ANSI X3.159) includes a number of library functions that modify their behavior according to locale. ANSI C also defines a number of multibyte functions and an additional function for manipulating monetary values. Open Software Foundation (OSF) members are in the process of specifying the internationalization requirements for OSF/1 Operating System Component (OSC). The major thrust, however, has been in the X/OPEN Group.

6.3.1.1 X/OPEN Internationalization

The X/OPEN Native Language System (NLS) was first published in the X/OPEN Portability Guide (XPG-2), was refined in XPG-3, and is being considered for revisions in XPG-4. NLS defines facilities for the development of internationalized applications that use 8-bit coded character sets. NLS provides the following facilities:

- Message catalogues, to allow program messages to be separated from the program logic, translated into different native languages and retrieved by the program at the time of execution
- An announcement mechanism, whereby a locale appropriate to each user can be identified to applications at the time of execution
- Internationalized C library functions, to provide a facility for locale specific processing
- A set of library functions, to allow the program to dynamically determine data specific to a culture or language
- Regular expressions, to extend the standard provisions by providing for the specification of locale classes, accented base characters, and multicharacter collating elements
- A set of standard commands, to provide 8-bit transparency for the processing of data and the name of file storage objects

Under consideration for XPG-4 is the question of general multibyte character support. Proposals have been advanced to require full internationalization of the command set, including multibyte handling. Digital has advanced proposals for handling multidirectional text and for providing locale as an explicit argument to NLS operations.

OZIX is XPG compliant. With regard to internationalization, OZIX contains message catalogues that are compatible with NLS, and provides the NLS facilities listed above.

6.3.1.2 Multiple Octet Character Set

The Multiple Octet Character Set (MOCS) is a proposed standard for a worldwide character set, and is currently under consideration in ISO-IEC JTC1/SC2 as ISO/IEC JTC1 10646 DP. MOCS proposes mapping the characters of all existing character sets into a single character space populated with four-*octet* characters (an octet consists of eight bits). Digital is actively supporting the MOCS effort, which is expected to become a published standard by 1992.

MOCS is a character set definition that extends the concept of 1-byte character encoding arranged as a linear array to four-octet characters. MOCS solves the problems arising from mixed character sets by containing all characters in a single character set. MOCS proposes a 1,2,3 and 4-octet usage, with the possibility of setting the usage form via a self-announcement mechanism. The default is equivalent to the 8-bit ISO Latin-1 character set to provide backwards compatibility.

MOCS does not solve the problem of how to handle character strings that contain characters of different writing directions. For instance, right-to-left text may contain embedded left-to-right elements. When only MOCS is used for strings containing mixed writing directions, the writing must be derived implicitly from the characters themselves. Implicit derivation of writing direction, unfortunately, does not always lead to an unambiguous rendering of the text string.

OZIX provides library routines to perform conversions between MOCS and other character sets. OZIX uses strings of fully expanded MOCS characters (four octets) to store identifiers for such things as file names and device names. The identifier strings are not simple, but rather are of compound string type.

6.3.1.3 Compound String

A *compound string* is an opaque string type used to represent both the character values in string data and the attribute information needed to correctly process the string data. Compound strings are based on the Digital Data Interchange Format (DDIF) and Digital Data Interchange Syntax (DDIS), which are based on the ISO ASN.1 notation.

The attributes recorded in compound strings are mainly required to correctly render the characters of the string for human presentation. For example, attributes include specification of:

- Character set
- Language
- Writing direction

Compound strings also allow for string data of mixed character sets, language, and writing direction. These attributes are the minimum required for full internationalization support of languages such as Hebrew, Chinese, Korean, Japanese, and so on. The compound string format is easily extensible without change to the application program interface, allowing for the future addition of alternate representations of the string, including voice, graphics, and so on.

A compound string is not a simple linear array of characters; rather, it is a nested hierarchy of string segments and requires library support to be manipulated in all but the simplest manner. It is therefore potentially more costly to process compound strings (in terms of processor cycles and storage requirements) than either MOCS or most other simple character set-based strings. However, no model based on character sets alone provides the flexibility or future extensibility offered by compound strings.

Compound strings are supported in the DECwindows Toolkit as the sole interface to text string arguments, and are used in the declarations of text string resources in UIL widget definitions. They have been proposed to X/OPEN I18N Working Group as the mechanism to handle mixed writing directions.[1]

OZIX stores file storage object identifiers such as file names or directory names in compound strings containing four-octet MOCS characters. In addition, OZIX supplies a library of compound string entry points to provide standard I/O services plus additional library routines to manipulate compound strings or perform conversion between compound strings and simple strings.

[1] See the paper *Proposal For Mixed Writing Directions Support* by Mike Feldman, IED (June 5, 1989).

6.3.1.4 Producing International Products (PIP) International Product Model

The OZIX product model follows the guidelines for creating international software products as described in the *Producing International Products (PIP) Reference Set*. The PIP divides its international product model into four components:

- International Base Component (A Component)
- User Interface Component (B Component)
- Market-Specific Component (C Component)
- Country-Specific Information Component (D Component)

The A Component is the invariant part of OZIX, and can be sold worldwide without modification. Major elements of the A Component in OZIX are the base system, the file system, the messaging facility, and parts of the terminal services.

The B Component is the language and text processing component of OZIX, and must be localized to the language and cultural requirements of a specific local market. Major elements of the B Component in OZIX include message text, language specific processing in the terminal services, online help, documentation, and the user interface to OZIX components such as system management or diagnostics.

The C Component is an element added to the A Component to meet special requirements of a specific market, such as specialized text processing software for Japanese text. The C component extends the A Component without requiring any changes to the A Component.

The D Component is a set of mandatory documentation, packaging and labeling, warranty, support and product description information added to meet all regulations required to sell OZIX in a specified country. The D component places no special requirements on the design of the other components.

6.3.1.5 Unified Text Representation (UTR)

The Unified Text Representation (UTR) architecture has been proposed by Ron Brender as an approach to handling multiple character sets in Digital's software systems.[2] The main assertions of the UTR architecture are:

- MOCS is used as the primary character set throughout the system
- For files that are simple text, a file attribute is used to specify the text representation used in each file
- A file processing attribute is used by a program when opening a file to specify the representation wanted for processing
- Text is automatically converted (in both directions) between the two representations when they are different

OZIX extends UTR by using compound strings for system identifiers. The notion that pure text files can be tagged according to the character data contained within them is recognized in the OZIX system design and utilized by OZIX libraries to perform limited, automatic text conversions during file access between compound string files and 8-bit character data files.

[2] See the paper *Unified Text Representation (UTR): A System Overview* by Ron Brender, ABSS, (August 18, 1989).

In addition to UTR, a number of engineers from Asian Base Systems Software (ABSS) and IED have been working on a model for internationalized software, and have published their findings.[3] OZIX follows their recommended model, which is essentially UTR extended to use compound strings, with MOCS as one of the supported character sets.

6.4 OZIX International Components

This section describes the technology present in OZIX components to support execution of both localizable software and multilingual software.

6.4.1 Commands

An international command is a user level program that functions correctly regardless of the user's natural language. At a minimum all OZIX commands provide 8-bit transparency for the processing of data and the names of file storage objects such as file names or directory names. In addition, all OZIX commands use international message and language switching facilities, thus accommodating easy translation by local engineering groups. Some selected OZIX commands are fully internationalized and have the capacity to process data and file storage objects presented in character sets other than the ISO 8859-1 character set (8-bit Latin-1). OZIX internationalized commands use the OZIX compound string library routines to achieve this language neutrality.

\The selection of commands to internationalize is incomplete at this time. Selection is a joint decision process between marketing and engineering groups.\

6.4.2 Libraries

Internationalization support in OZIX libraries is contained in standard C libraries, compound string library extensions, and the compound string I/O library.

OZIX supplies a set of standard C libraries to provide string handling routines to deal with string parameters made up of 8-bit characters. Existing C applications link against the OZIX standard C libraries and enjoy complete compatibility.

OZIX extends the standard C library with an additional set of library routines that correspond to standard C library routines dealing with system identifiers. The compound string library extensions are used by new international applications that wish to be language neutral in expressing file names, directory names, user names, etc. Compound string library extensions depend on underlying operating system support for handling identifiers presented in compound strings.

Finally, OZIX provides a library of compound string I/O and utility routines that are used by new international applications. Utility routines manipulate compound strings and do conversions between compound strings and simple character strings. Compound string I/O routines parallel the standard I/O routines, except that the strings manipulated or processed through input and output are of compound string type rather than simple strings. The compound string I/O library is designed so that its routines are available to all languages supported in OZIX. It is a long-term goal that the compound string I/O library can be ported to other open systems.

Automatic conversion between compound string text and 8-bit text may occur in OZIX libraries. The key to library conversions is the concept of a data attribute to describe the character set contents of OZIX text files. For example, if an application uses the standard C library to access a text file that is tagged as compound string, the standard C library routines automatically convert compound string text data to eight-bit text data for delivery to the application. Likewise, if an application uses

[3] See the paper *Digital's Text Model for the Future* by Jürgen Bettels, Ron Brender, Tim Greenwood and Jim Saunders, August 22, 1989.

the compound string I/O library to access an 8-bit text file, the compound string library routines automatically convert 8-bit text data to compound string text data for delivery to the application.

Please note that no other types of conversion take place other than these two examples. In addition, file data attributes that describe character set contents may be examined or modified through a set of existing utilities that have been extended to understand file data types.

6.4.3 Terminal Services

The OZIX terminal subsystem provides support for multiple character sets and languages. The terminal subsystem is structured in accordance with the PIP A-B-C-D model, and allows for the inclusion of additional locale-specific character set handling without modification of the basic terminal subsystem code or data structures. To facilitate easy exchange of character data, the terminal subsystem uses the four-octet MOCS format internally for some of its character data buffers.

The primary interface to the terminal subsystem for an existing application is the OZIX OSF API. The terminal subsystem supports POSIX style functions such as *read* or *write*. In addition, the terminal subsystem supports communication of character data (as opposed to byte strings) through the OZIX compound string library interface.

6.4.4 Base System

The OZIX base system components are designed to provide language-neutral support for applications. Base system components are structured in accordance with the PIP A-B-C-D model by isolating message text to message catalogs, etc.

Base system components use four-octet MOCS characters within compound strings as the format for storing file storage object names such as file names, directory names, or device names, and for other system identifiers such as user names.

6.4.5 Messaging

The OZIX messaging facility is responsible for defining condition values and managing the association between a condition value and some short descriptive text (*message*). Condition values, system wide in scope, are associated with message text files that are created through the messaging facility. The data stored in message text files is retrieved at the time of execution through the use of message text routines furnished by the messaging facility.

In concert with open system requirements, the OZIX messaging facility supports open system message catalogs as defined in X/OPEN NLS. In addition, the messaging facility is multilingual; that is, it has the capability of handling multiple character sets and of supporting multiple concurrent locales. Pervasive use of the messaging facility throughout OZIX facilitates translation of components by separating message text from functional code.

6.4.6 Programming Tools

The OSG C compiler that is bundled for shipment with OZIX provides support for the development of international applications. The C compiler provides multibyte handling in source code comments, strings literals, and character constants. In addition, the C compiler provides wide character handling. It is the goal of the OSG C compiler to provide the support necessary for OZIX to be in compliance with the latest X/OPEN branding requirements.

Compound string support is provided in the OZIX run-time libraries. All development tools, such as the compiler, loader, and debugger, also have compound string handling capability.

6.4.7 File System

The OZIX file name subsystem is responsible for maintaining the hierarchical file namespace. The file name subsystem provides path traversal services and directory services to its clients.

The file name subsystem uses four-octet MOCS characters within compound strings as the format for storing file names, as well as other types of strings. As with other basic OZIX components, the file name subsystem is designed to provide language-neutral support for applications, and is structured in accordance with the PIP A-B-C-D model by isolating message text to message catalogs, and so on.

A file attribute to describe the character set contents of a file is maintained for each file. Each system maintains a default to be used for creating new files, when none is explicitly specified for file creation.

6.5 Additional Internationalization Support

Technology alone cannot guarantee that OZIX will be a successful international product. Other support required for success in the international market is described in this section.

6.5.1 Documentation

The OZIX documentation set is designed to complement the international OZIX product. In order to optimize translation to other languages, the entire information set is organized in accordance with the guidelines described in the PIP and in the *Planning for Translation* document. In recognition of the needs of different kinds of users, the information set is structured in a modular and hierarchical fashion to allow maximum translation flexibility. Local Engineering Groups may translate only selected information and structure that information as required to meet the needs of the local language and culture, without rewriting the original user information. Finally, stressing commonality of information across the entire ULTRIX family lessens redundancy in translation efforts.

Documentation tools also play a supporting role in the success of the international OZIX product. Online documentation tools for OZIX support the creation and display of European, Asian, and Semitic languages as well as mixed writing direction. Online documentation tools also support concurrent multiple languages, where the primary, secondary, and default language can be specified.

6.5.2 OZIX Release Strategy

A key part of the OZIX international product strategy is the configuration and coordination effort involved in creating the OZIX release kit. Three types of components create an OZIX system tailored to local market requirements:

- A common core system developed by the OZIX product team, combined with international Digital bundled and layered products and third party products.
- Country and language specific components, such as translated messages, language specific collating sequences, or a localized spelling checker library.
- Country-specific added value, such as tools tailored to a specific language or method of presentation.

At system installation, an OZIX kit localized for a specific market may be augmented by other country and language specific components. This augmentation allows a customer whose native language is Japanese, for instance, to produce applications suitable for other Far Eastern markets.

6.5.3 Training, Support and Service

Three other areas provide internationalization support to make OZIX a successful international product: training, support, and service. All three areas deal with the multilingual environment. The training curriculum for OZIX uses a variety of format types to best meet the learning and delivery needs of customers throughout the world. In order to support systems containing more than one local environment, services are developing a strategy for support of worldwide products and working to ensure that service personnel have access to appropriate resources. Finally, support personnel will provide expertise to help the customer develop international products or applications.

CHAPTER 7

BASE SYSTEM ARCHITECTURE

7.1 Introduction

The OZIX Base System Architecture defines the functional and conceptual primitives used to design the OZIX operating system.

Instead of defining a system consisting of a single, large, monolithic kernel, the architecture defines individual modules called *subsystems*, supported by a tiny "kernel" called the *nub*. Subsystems are the "building blocks" of the OZIX design.

A subsystem is a body of instructions and read, or read/write, data that provides a set of functions and data abstractions. Each subsystem provides functions that can be invoked by other subsystems in the OZIX operating system, which includes user applications. User applications themselves are implemented using subsystems; although from the user's perspective, they behave exactly like POSIX user processes.

7.2 OZIX Base System Architecture Goal

The goal of the OZIX base system architecture is to foster a crisply defined modular system design that achieves the aggressive security, integrity, and performance goals of the OZIX operating system.

7.3 Base System Architecture Structure

The OZIX base system architecture defines two types of functional components: subsystems and the nub. Since an operating system cannot be defined by function alone, the virtual memory environment and execution of subsystem functions are defined by two *models*: the *execution* model and the *virtual memory* model.

The two types of functional components and the two conceptual models that make up the architecture are briefly summarized below, then discussed in detail in the following sections.

The two functional components of the architecture are:

- Subsystems

  Subsystems are the modular functional units of the base system architecture. Each subsystem serves a specific role, such as file system, memory management, or device driver support. Subsystems provide the basis for controlled isolation in the operating system design and implementation. This separation of subsystems is in turn the basis for the integrity and protection of the entire system.

  Subsystems are also the basis for extending and evolving the OZIX operating system in the future. Whenever a new OZIX system capability is to be made available, a new subsystem can be incorporated, thus extending the operating system itself.

- The Nub

  As its name is meant to imply, the nub represents a very small part of the overall system code. The nub contains the primitive functions necessary to efficiently execute the code contained in subsystems. As a result, the nub contains most of the code that is dependent on the specific processor and hardware virtual memory architectures.

The execution and virtual memory models define how subsystems and the nub relate to physical memory and the processors to perform useful work.

- Executor Model

  The executor model defines how user and system computational tasks are described and supported by the OZIX subsystems.

  An *executor* consists of a family of threads executing code to perform computational tasks which may be implemented in one or more subsystems. In OZIX, the traditional "user process" is simply an executor originating in an application subsystem.

  The security and accounting identity of an executor is established at the time of its creation. An executor's threads maintain this same security and accounting identity as they pass through the various subsystems.

- Virtual Memory Model

  The virtual memory resources used by executors are described by the virtual memory model. The virtual memory model presents memory in terms of *memory segments*, which represent subsystem code and executor data.

  The primary goal of the virtual memory model is to allow the precise specification of an executor's access to, and use of, virtual memory. The virtual memory accessible by an executor while it is executing code in a subsystem represents a unique view of memory. This subsystem/executor-specific view of memory is referred to as a *subsystem execution context*, or *SEC*.

  The virtual memory model isolates the hardware-dependent CPU and memory management details from the subsystem designer. This helps make the entire system less dependent on specific CPU architectures.

7.4 OZIX Base System Architecture Functional Modules

All of the code in the OZIX system is executed in either a subsystem or in the nub. The vast majority of code is executed in subsystems, as the nub contains only the minimum functions necessary to support the virtual memory environment and execution of code within a subsystem.

Common procedures used by many subsystems may be implemented by code libraries. When such library procedures are called by an executor in a given subsystem, they execute in the same SEC as the subsystem code that called them. For this reason, a library procedure is considered part of a subsystem's code from the perspective of the architecture.

Subsystems and the nub are described in Section 7.4.1 and Section 7.4.2, respectively.

7.4.1 OZIX Subsystems

Each OZIX subsystem supports a documented interface through which specific functions and data structures can be accessed by executors from other subsystems. OZIX subsystems can be thought of as object oriented in the sense that they can export protected objects, as well as operations that can be performed on those objects.

A subsystem can be described from two perspectives:

- External Perspective - The subsystem as viewed by the caller
- Internal Perspective - The resources used by the subsystem on behalf of the caller

The external perspective of a subsystem is nothing more than its procedural interface; that is, the functions it can perform on behalf of executors. An executor in one subsystem can invoke a procedure in another subsystem by issuing a *subsystem procedure call*. Subsystem procedures are discussed in Section 7.4.1.1.

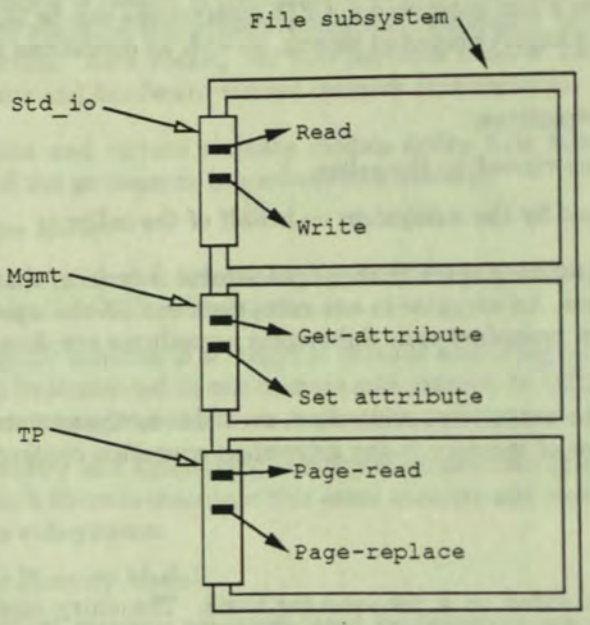
The internal perspective of a subsystem is the execution environment available to the executor while executing a subsystem's functions. This view of memory is the subsystem execution context, which is explained in Section 7.4.1.2.

7.4.1.1 Subsystem Procedures

Access to a subsystem's procedures are controlled on a per-executor basis. The entry points to a subsystem's procedures can be grouped into *packages*, which control access by a given executor to the group of subsystem entry points. By grouping subsystem entry points into packages, an executor, for example, may be granted access to all of the procedures in the "service" package of a subsystem, but denied access to the procedures in the "management" package.

Subsystem procedures are named hierarchically. Any subsystem procedure in OZIX has a name of the form *Subsystem_name.Package_name.Procedure_name*. This way a common *File_io* package can be defined, even though many subsystems may implement that package.

Figure 13: A simple subsystem



A functional view of a subsystem is illustrated in Figure 13. This simple example shows the "File" subsystem implementing three packages of two procedures each. The "Std_io" package implements support for the standard OSF "Read" and "Write" functions. The "Mgmt" package implements the standard OZIX system management "Get-attribute" and "Set-attribute" procedures. The "TP" package implements value-added "Page-read" and "Page-write" procedures for use by transaction-processing database subsystems.

7.4.1.1.1 Subsystem Procedure Calls (SPC)

Subsystem procedure calls, or SPCs, are used by an executor in one subsystem to invoke a specific package entry point, or procedure, in another subsystem. From the perspective of the caller, a subsystem procedure call is identical to a normal procedure or system call. In order to better understand an SPC, first consider the essential steps involved in a normal procedure call:

1. Code in the caller assembles the call parameters, or argument list, according to the calling standard. The calling standard typically specifies registers for some number of arguments, and stack local storage for the remainder.
2. Some form of linked jump is performed, which saves return information and transfers control to the call site, or called procedure.
3. Code at the call site will typically save registers used by the called procedure, possibly perform other bookkeeping or optimization operations, and then begin execution of the procedure body.
4. Control is returned back to the caller through the saved jump linkage.

An SPC is nearly identical. The basic steps are:

1. Assemble call parameters in the caller in exactly the same manner as the normal procedure call above.
2. Execute a protected transition to the called procedure while changing to the SEC of that procedure's subsystem.

3. Execute the procedure code at the called site in the same manner as a normal procedure call.
4. Execute a protected transition back to the caller, restoring the caller's SEC.

7.4.1.1.2 Gates

The difference between a normal procedure call and an SPC is the linkage mechanism used to transfer control from the caller to the called procedure, and back to the caller. This mechanism, used in Steps 2 and 4 above, is referred to as a *gate*. The protected transition implemented by a gate is referred to as a *gate crossing*.

The gate crossing mechanism is designed to operate with as little overhead as possible. In particular, no access or security checks are performed by the gate mechanism itself, which is only responsible for executing a protected transition from one SEC to another SEC. Any necessary access checks are made when an SEC is constructed, or possibly deferred until the first reference is made to some memory object in the SEC.

Likewise, the gate does nothing with the procedure call argument list beyond making the arguments available in the called procedure's SEC. In particular, the gate does not track down indirect references, such as those represented by pointers. This ensures that the gate is kept simple enough to provide maximum performance.

7.4.1.1.3 Implicit and Explicit Subsystem Procedure Calls

Subsystems are dynamically loaded into the OZIX system. As a consequence, calls to subsystem procedures are dynamically resolved when they are initially executed, much like calls to shareable library routines are resolved by autoloading code when first called.

An SPC can be made either implicitly or explicitly. An implicit SPC is coded in the same manner as a call to an external routine. Compiler, linker, and autoload routines create the illusion of a normal procedure call using shared library mechanisms.

An explicit SPC uses the symbolic subsystem procedure name resolution and gate crossing mechanisms directly when subsystem calls can only be resolved at runtime by the calling subsystem. It is comparable to calling a function through a pointer in traditional systems. Examples of subsystems that require this dynamic call resolution capability are:

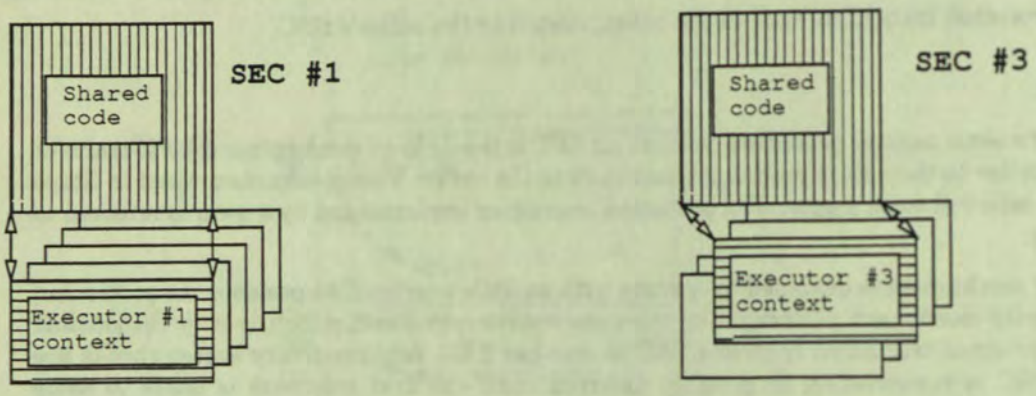
- A subsystem which dynamically resolves references to file and device subsystems as part of pathname traversal
- Network subsystems, which must dynamically resolve references to various transport and routing subsystems for a specified protocol stack

7.4.1.2 Subsystem Execution Context

A subsystem execution context is the set of memory segments accessible to an executor within a given subsystem. Figure 14 illustrates a number of key points about subsystem execution context.

Both figures in the illustration show a subsystem with shared code and executor-specific context. In the left hand figure, the SEC for executor #1 is highlighted. In the right hand figure, the SEC for executor #3 is shown. In both cases, the same shared code segment is accessible. The difference is simply which executor-specific segment is mapped.

Figure 14: Subsystem Execution Context



There are many design possibilities between the strictly executor-specific memory segments and completely shared segments illustrated. The degree of sharing is determined by the subsystem designers and controlled by the access control associated with the logical memory segment.

7.4.2 OZIX Nub

The nub provides the primitive functions required to execute subsystem code. The functions provided by the nub are those that require execution of privileged hardware instructions, access to physical memory, visibility across address spaces, or those functions that occur in special hardware states. Nub functions include thread scheduling and synchronization, CPU management, subsystem gate crossing, interrupt dispatching, and fault detection primitives.

The OZIX nub is a small piece of software, since most of the traditional kernel-level functions are implemented by the subsystems. In order to simplify the task of migrating to different hardware platforms, the nub is structured as two subcomponents, one hardware-dependent and the other hardware-independent.

The nub code executes in a hardware protected mode, such as kernel mode. The nub must execute in a protected execution context in order to have the capability to make the protected transitions between subsystems, which execute in separate virtual contexts.

The nub supports fault detection and recovery by detecting both hardware and software faults and initiating global fault recovery through the fault recovery subsystem.

7.5 OZIX Base System Architecture Conceptual Models

As discussed earlier, subsystems and the nub operate within the framework of two basic architectural concepts: the executor model, the virtual memory model. Both of these models are discussed in detail in the following sections.

7.5.1 Executor Model

An executor, loosely, is a family of threads working together on a common task. A thread represents a machine state, an execution stack, and a security and accounting identity. More precisely, then, an executor is defined to be a set of threads that share the following attributes:

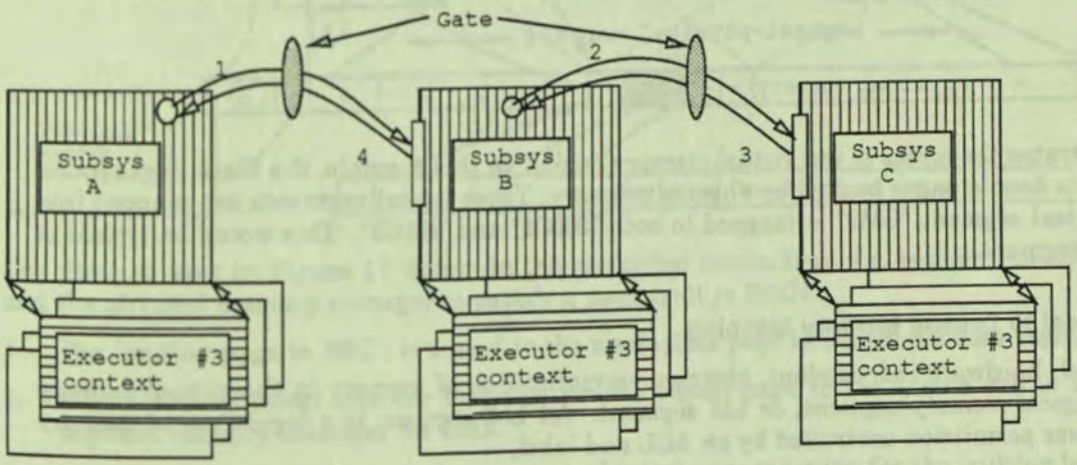
- User Identity
- Security and Integrity Labels

- Accounting ID
- Set of Subsystem Execution Contexts

Figure 15 illustrates a single executor, identified as "#3", making an SPC from subsystem "A" to subsystem "B", which in turn makes an SPC to subsystem "C", then returns. In this example, each subsystem execution context contains executor-specific context, as shown by the "Executor #3 context" memory segment in each subsystem. These executor-specific segments are private to each subsystem. Note, however, that it is the same segment (and contents) whenever any thread of executor #3 is in a given subsystem.

An executor executes "through" subsystems with a set of related and persistent subsystem execution contexts. The individual threads of a subsystem are scheduled for processor execution. The SPC gate mechanism itself does not cause any scheduling decisions to be made.

Figure 15: Executor SPC calls



There are many active, but distinct, executors. Some executors may represent customer applications, while others represent special system support functions. From an architectural standpoint, executors differ only in their task and one or more of the executor attributes listed above.

7.5.2 Virtual Memory Model

The OZIX virtual memory model builds an abstract virtual memory environment from the physical memory and virtual memory mapping features of the hardware. It is within this virtual memory environment that subsystem code executes. There are three layers of abstraction to the virtual memory model:

- Physical Memory
- Logical Memory Segments
- Subsystem Execution Context

The memory that is actually implemented by the hardware system is, of course, the *physical memory* available to the OZIX operating system. Abstract vectors of pages are known as *logical memory segments* (LM segments). LM segments describe the mapping of logical pages to physical pages. The dynamic reassignment of the limited number of physical pages to the much larger set of logical pages is a function of *physical memory management*.

A subsystem is composed of one or more logical memory segments containing executable code, and one or more logical memory segments containing data. The set of logical memory segments mapped to a specific executor while executing in a given subsystem is known as a *subsystem execution context*, or SEC.

Figure 16: OZIX Virtual Memory Model

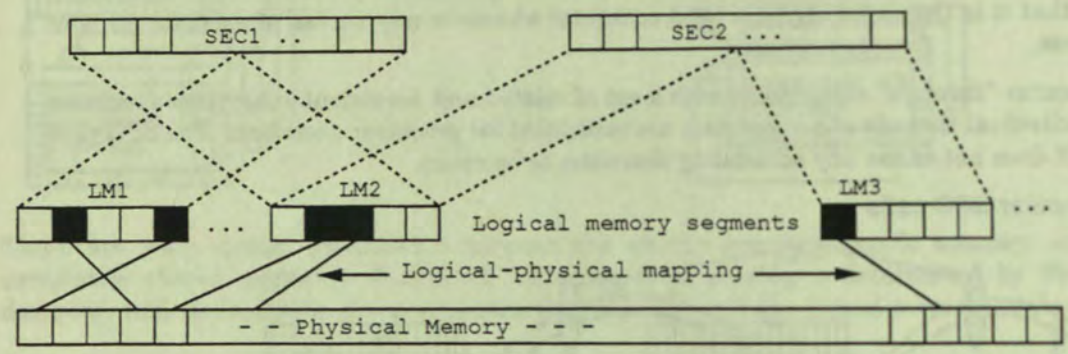


Figure 16 illustrates the layers of the virtual memory model. In this example, the black pages in the logical segments denote pages backed by physical memory. Three logical segments are mapped into two SECs. Logical segment "LM2" is mapped to both "SEC1" and "SEC2". This would be typical of a shared code segment.

7.5.2.1 Physical to Logical Memory Mapping

The lowest-level, hardware-independent, abstract representation of memory in the virtual memory model is the *logical memory segment*, or LM segment. An LM segment is a logical set of memory pages with access permission controlled by an ACL and label.

The pages in an LM segment are themselves mapped to physical memory resources in a hardware-dependent fashion. LM segments define one of the fundamental hardware dependent/independent boundaries in the OZIX system architecture. LM segments insulate memory managers from the idiosyncrasies of the particular hardware implementation represented by the OZIX physical memory management software. For example, in a paged machine, an LM segment may contain the mapping information and a page table.

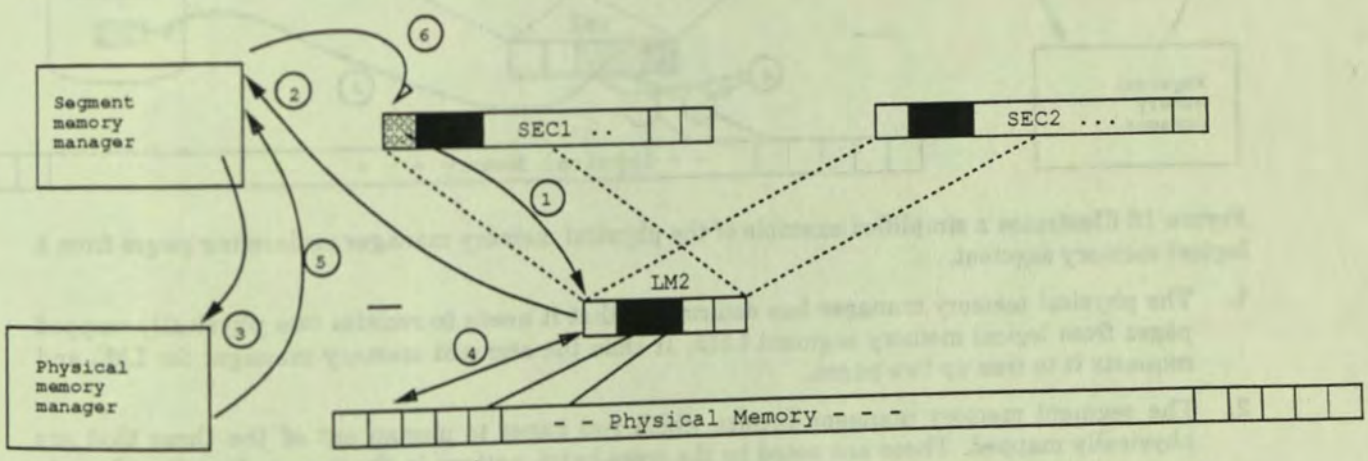
7.5.2.2 Physical Memory Management

OZIX physical memory management uses LM segments as the basis for physical memory resource accounting and management. The OZIX physical memory management software maintains the state of the LM segment mapping information. In a way, physical memory can be viewed as a "cache" of LM segment content. In general, the physical memory management software attempts to remove physical memory mappings from inactive logical segments so the physical memory can be reassigned to active logical segment pages.

The OZIX virtual memory model defines a very flexible and general mechanism. Simply stated, each LM segment has a *segment memory manager* associated with it. The segment manager for a given LM segment implements the "inpage" and "outpage" algorithms for the logical pages in that segment. This allows paging algorithms to be precisely tuned for different uses of virtual memory.

Another way to view this is that segment memory managers negotiate with the physical memory management software to add and delete physical memory for their LM segments. The segment memory manager makes requests to the physical memory manager for physical pages to satisfy page faults for segment pages, and the physical memory manager makes requests to the segment memory manager to reclaim inactive physical pages for use by the actively faulting segments.

Figure 17: Logical segment page fault resolution



The directed arcs in Figure 17 illustrate the simplified interaction of a segment memory manager and the physical memory manager to satisfy a page fault in SEC1.

1. The faulting page in SEC1 is traced to the appropiate page in logical memory segment LM2.
2. There is no physical memory mapping for the logical page in LM2, so a call is made to the segment memory manager for LM2.
3. The segment memory manager requests a physical page mapping for the faulting logical page.
4. The physical memory manager maps a physical page to the desired logical page.
5. Control returns to the segment memory manager with a valid virtual mapping for the faulting page. The segment memory manager must now initialize the page appropriately. For example, the segment memory manager might read a page from a file if it is implementing a "mapped file" type of segment.
6. After the segment memory manager has successfully initialized the page, it returns control to the virtual memory fault handler, which restarts the faulting instruction.

Figure 18: Physical memory page reclamation

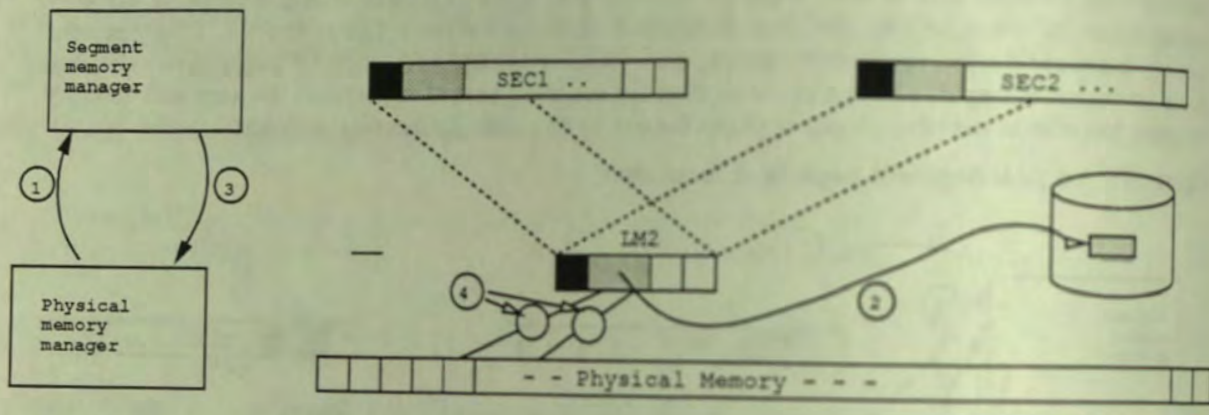


Figure 18 illustrates a simplified example of the physical memory manager reclaiming pages from a logical memory segment.

1. The physical memory manager has determined that it needs to reclaim two physically mapped pages from logical memory segment LM2. It calls the segment memory manager for LM2 and requests it to free up two pages.
2. The segment memory manager decides which two pages to unmap out of the three that are physically mapped. These are noted by the cross-hatch pattern in the figure. It writes them to mass storage so they can be recovered later.
3. After successfully saving the two pages on mass storage, the segment memory manager returns control to the physical memory manager.
4. The physical memory manager breaks the physical mapping for the logical pages specified by the segment memory manager.

The significant features of this memory management model are:

- There can be many different logical page management algorithms and policies beyond standard file paging. They can be tailored to satisfy specific requirements, such as a data buffer cache.
- Physical to logical management is simple and direct. When the physical memory manager decides it wants a page back from an LM segment, it gets it.

7.6 Summary of OZIX Base System Architecture Features

The OZIX base system architecture defines a truly modular system for designing and implementing an operating system. Key distinguishing features of this architecture are:

- Subsystems are used to implement essentially the entire system. Subsystems define independent, protected execution contexts, which provide strong data encapsulation capabilities to system designers. This is in marked contrast to typical commercial operating systems, in which the entire system operates in a very small (at best) number of protected execution contexts.
- A memory model which allows different memory management algorithms to efficiently compete for common physical memory resources.
- An execution model, which separates scheduling decisions from execution environment transitions, as well as defining the primitive unit of identification in the security model.

The OZIX base system architecture creates a *real* modular framework upon which the OZIX system is built. As such, it represents a significant step beyond the use of modular design techniques. It creates profound opportunities for elegant and innovative design solutions to the perennial operating system problems of reliability, security, integrity, extensibility, flexibility, and continuous operation.

CHAPTER 8

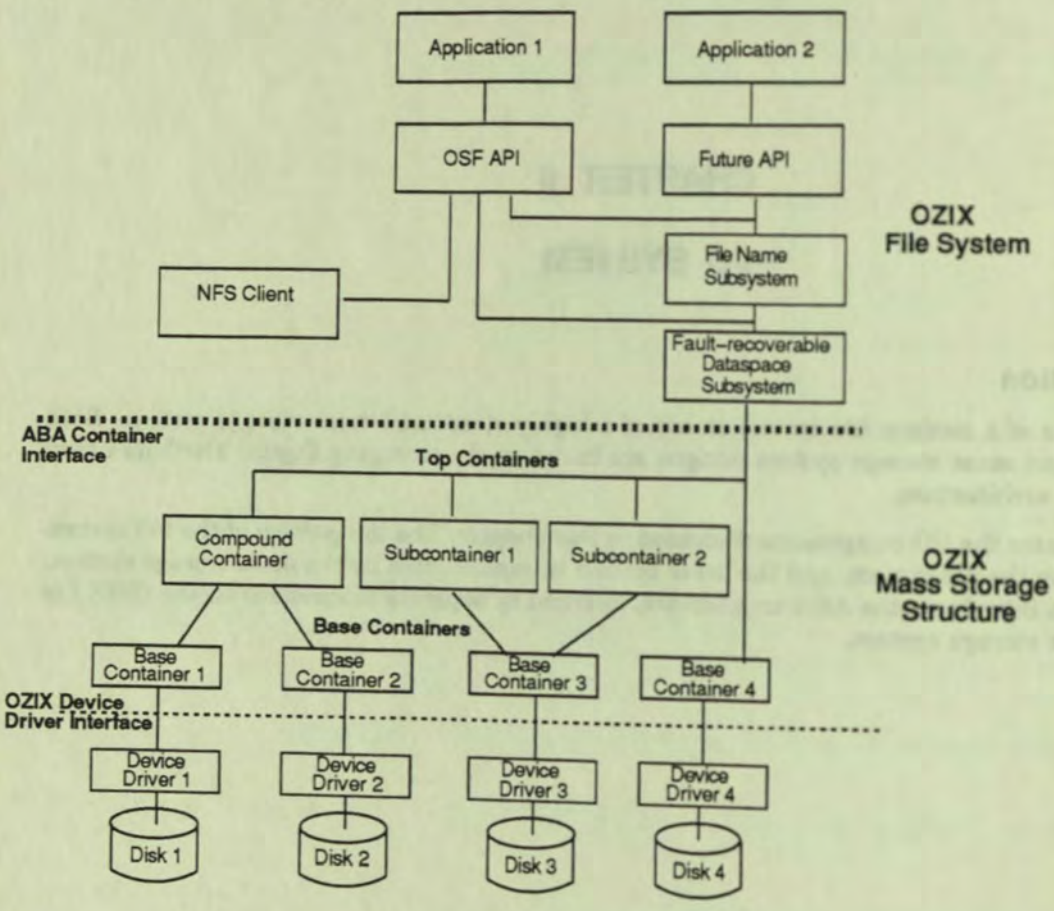
I/O SYSTEM

8.1 Introduction

OZIX I/O consists of a modern file system on top of a highly structured mass storage system. Both the file system and mass storage system designs are based on the emerging Digital attribute-based allocation (ABA) architecture.

Figure 19 illustrates the I/O components discussed in this chapter. The top portion of the I/O system is implemented by the file system, and the lower portion is implemented by the mass storage system. This chapter first introduces the ABA architecture, followed by separate discussions on the OZIX file system and mass storage system.

Figure 19: Overview of OZIX I/O System



8.1.1 Goals

The goals of the OZIX file system and mass storage designs are as follows:

File System Goals

The main goals of the OZIX file system design are to:

- Create a faster, more robust file system that is accessible through the POSIX 1003.1 file system interface. The file system design exploits transaction processing techniques to guarantee seven-days-a-week/twenty-four-hours-a-day (7/24) reliability, minimize response time, and optimize the updating of on-disk data structures.
- Design the OZIX file storage architecture to handle very large disk farms. This capability is made possible by developing the *attribute-based allocation* (ABA) concept into a workable architecture. The ABA architecture will initially be implemented on OZIX, and may be implemented on other operating systems in the future.
- Extend the POSIX interface by adding resource manager support services to support data base systems in a transaction processing environment.
- Support the OZIX B2 security design. Dataspaces are objects in the OZIX security system, and as such have sensitivity and integrity labels, and are subject to mandatory access and discretionary access controls.

- Support Hierarchical Storage Management (HSM) for better utilization of storage devices. Devices are organized in a hierarchy of faster and slower devices, and HSM migrates files between devices.
- Create a filename subsystem that can be called by multiple file system interfaces. Initially a POSIX file system interface will be developed, but additional file system interfaces—such as Apple® AFP, and MS-DOS SMB—could be developed.
- Provide a media failure recovery mechanism for fast and reliable dataspace recovery from media failures.

Mass Storage Goals

The major goals of the OZIX mass storage design are:

- Provide the basis for very high I/O throughput at very low latency. A set of ABA mechanisms support highly variable hardware topologies where available hardware capability can be applied where it is most needed.
- Support both DSA and SCSI devices. The design will also be able to support future storage technologies, interconnects and peripherals.
- Provide the capabilities to create storage abstractions, such as disk striping and shadowing, from the underlying storage hardware configuration and interconnects. Such storage abstractions are implemented as ABA abstractions on top of elemental devices.
- Support many different types of mass storage devices—including solid state devices (SSD), magnetic and optical disks, magnetic tapes, and CDROMs – by implementing drivers for the various device types.
- Provide for a robust I/O configuration management by implementing an I/O configuration manager based upon the Enterprise Management Architecture.
- Provide the capability of being extensible by designing the mass-storage components that takes full advantage of features provided by the OZIX Nub and other subsystems, like the memory management subsystem and the process subsystem. Mass-storage components will be designed to be adaptive so that the components operate optimally, within the bounds of the system resources.

8.2 Attribute-based Allocation Architecture

The overall goal of the ABA architecture is to make the specifics of allocating and managing storage media transparent to the file system or application. The ABA architecture is designed to separate file storage allocation from device location by presenting higher levels of software with an abstract view of mass storage.

ABA effectively manages a multi-level hierarchy of differing storage devices, according to a set of policies. Using ABA, it is possible for storage management systems to provide 7x24 service for very large file and database systems with minimal impact on running applications. Using ABA, it will be possible for storage systems to deliver average throughput approximating that of the fastest data storage device with average costs and capacity approximating that of the least expensive storage device.

The ABA architecture is made up of a *dataspace* layer and a *container* layer. Dataspaces are implemented by the file system; containers are used in the OZIX mass storage system.

Dataspaces

The ABA dataspace layer represents mass storage media as being made up of logical units called dataspaces. A dataspace is a virtually addressed storage space to which data is written and from which data is retrieved. A dataspace may represent part of a file, a file, or several files. Each dataspace is associated with specific attributes that define its storage requirements, size, access characteristics, and so on.

Containers

The container layer of ABA presents all mass storage devices as *containers*, which are logical representations of the devices. Containers representing physical devices are known as *base containers*. Base containers may be partitioned into *subcontainers*. Containers that represent virtual device abstractions required for such functions as striping and shadowing are known as *compound containers*. Compound containers can also be partitioned into subcontainers.

As dataspaces are associated with attributes, each container is associated with specific properties. For example, a base container would have properties that indicate details about the underlying device, such as its speed, size, reliability, and so on.

Dataspace Migration

Instead of allocating a specific location on a specific disk, dataspaces are allocated in a container having the properties best suited for the particular file. Dataspace migration can occur because a user changes a file's attributes, the system manager requests a migration, or because storage management detects discrepancies between the storage resources desired and those that are available.

OZIX implements numerous migration strategies, such as:

- Migrate dataspaces to cheaper containers.
- Migrate dataspaces to high performance containers at a particular time of day when heavy use is anticipated.
- Migrate dataspaces within the same container when they become fragmented, thus consolidating them in contiguous container blocks.

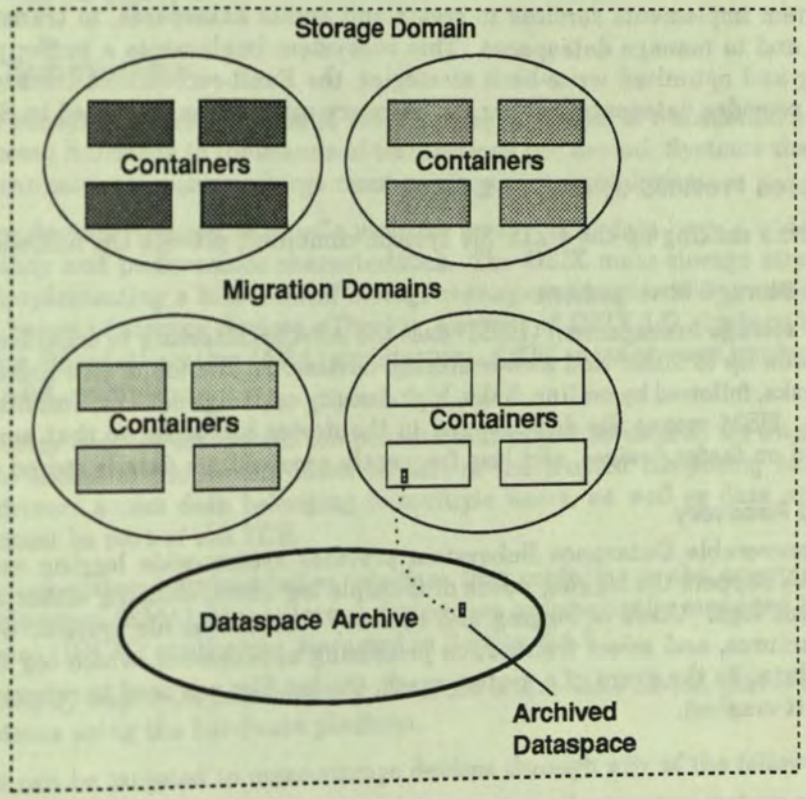
As illustrated in Figure 20 dataspaces migrate within a specified *migration domain*. Dataspaces may also be archived, in which case the dataspaces move out of their native migration domain into a *dataspace archive*. The dataspace archive is a repository for dataspaces that are not expected to be active for a long time. The entire storage domain of a system consists of one or more migration domains and one or more dataspace archives. Upon request, the dataspace from a dataspace archive can move back to its original migration domain.

8.3 File System

The OZIX file system is designed to provide quick and reliable access to files and data. Much of the design places special emphasis on fulfilling the needs of transaction processing (TP) resource managers, such as database systems. The functions used to support such resource managers provide the foundation for a superior general-purpose file system design and implementation.

The top portion of Figure 19 illustrates the OZIX file system. Note that only the OSF API is being implemented for V1 of OZIX.

Figure 20: ABA Storage, Migration, and Archival Domains



8.3.1 API Subsystems

The top level of the OZIX file system design is composed of API subsystems. These subsystems present a specific file interface to an application, such as the POSIX 1003.1 interface. OZIX is designed so that multiple APIs can be written to interact with the OZIX file system. Examples of APIs other than POSIX that could be developed are Apple AFP or MS-DOS SMB.

8.3.2 File Name Subsystem

The File Name Subsystem is the next layer of the OZIX file system. The file name subsystem presents a generic set of file and directory services that may be called by any API in OZIX. The File Name Subsystem provides directory and file services to the ABA-based files below the File Name Subsystem. Any API can, for example, open, read, write, and close an ABA-based file.

The File Name Subsystem also provides services to create mount points to mount objects other than local ABA files in the local directory hierarchy. These services, for example, allow remote directory trees that are accessed via NFS client services to exist in the local directory structure, and provide create and lookup routines for pipes and special files in the local directory structure.

8.3.3 Fault-recoverable Dataspace Subsystem

The Fault-recoverable Dataspace Subsystem implements the ABA architecture dataspace concept. This subsystem implements services to create and access dataspaces, to transfer data to and from dataspaces, and to manage dataspaces. This subsystem implements a buffer pool cache to provide read caching and optimized write-back strategies; the Fault-recoverable Dataspace Subsystem subsystem also provides dataspace logging and recovery services, as discussed in Section 8.3.4.

8.3.4 Features Provided by the File System

The subsystems making up the OZIX file system, combined, provide the following capabilities.

Hierarchical Storage Management

Hierarchical storage management (HSM) uses the ABA architecture to organize file data in a device hierarchy made up of faster and slower storage devices. At the top of this hierarchy there might be solid state disks, followed by on-line disks, high-density cartridge devices, and finally off-line magnetic tape devices. HSM moves file data around in the device hierarchy so that commonly accessed file data is stored on faster devices, and less frequently accessed file data is stored on slower devices.

Logging and Recovery

The Fault-recoverable Dataspace Subsystem provides system-wide logging and recovery services. These services support the logging needs of multiple log users, utilizing either a shared common log or independent logs. Users of logging and recovery include the file system, which logs updates to file metastructures, and select transaction processing applications, which log updates to important transaction data. In the event of a system crash, the log files are used to recover the system state at the moment it crashed.

Security

The OZIX file system supports discretionary access control list (ACL) security and non-discretionary secrecy and integrity controls on dataspaces.

Secrecy and integrity controls allow users to be segregated into different security levels and provide controls for restricting the access between those levels. Such controls are mandatory on B2 certified security systems.

Secrecy controls prevent users at lower levels from reading files maintained by users at higher levels and users at higher levels from writing files to lower levels.

Integrity controls keep a security breach that entered the system at a certain integrity level from moving to higher levels. Under these controls, users at lower levels are unable to write to higher levels, and users at higher levels are unable to read from lower levels. Such integrity controls represent significant added value for commercial systems.

8.3.5 Backup Strategy

OZIX provides two classes of backup technology to end-users. The first class consists of UNIX file backup utilities *tar* and *cpio* which provide standard mechanisms for file backup and data interchange among UNIX-based systems; there are not plans to support *dump/restore*. The main goal of this class is to provide compatibility with UNIX and UNIX standards such as X/OPEN and POSIX.

The second class provides Digital value-added backup and recovery features, which are able to handle the high-availability requirements of the large commercial systems which OZIX is intended to support. Since dataspaces and containers are the fundamental storage elements in OZIX, this class of backup is concerned with back up and recovery of dataspaces rather than files. Also, since the major goal

is to provide recovery from media failures, the selection of the dataspaces to back up is driven by the containers in which the dataspaces reside. This is in contrast to the file system and directory oriented backups done with *tar* and *cpio*.

8.4 Mass Storage Subsystems

The OZIX mass-storage subsystems are designed to meet the requirements of transaction processing (TP) systems, which process hundreds to thousands of transactions per second. Systems that process such large volumes of transactions require a large number of mass-storage devices.

The multitude of storage devices required to handle multiple terabytes of data have a wide range of cost, reliability, availability and performance characteristics. The OZIX mass-storage strategy provides opportunities for implementing a hierarchical storage management system that can efficiently support such a diverse range of storage devices. The key element of OZIX I/O strategy is the implementation of Attribute Based Allocation (ABA) architecture. OZIX mass-storage implements the *container* aspect of the ABA architecture.

As discussed in the Security chapter, all system components that might be used by an unauthorized user or program to gain access to the system must be part of the trusted computing base (TCB). Since most I/O device drivers access data belonging to multiple users, as well as data at multiple sensitivity levels, they must be part of the TCB.

OZIX has a comprehensive system administration interface that conforms to the emerging Enterprise Management Architecture (EMA). Mass-storage devices are automatically managed by the I/O configuration management (IOCM) application discussed in Section 8.4.3.

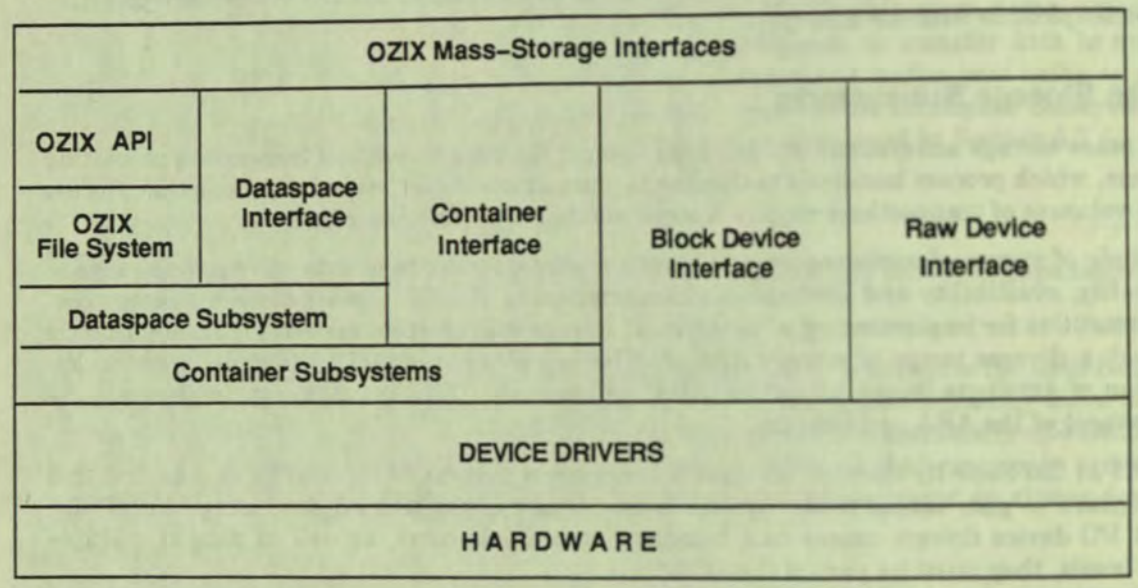
OZIX system will be booted by way of the code resident in a ROM or a console device, that is consistent across all operating systems using the hardware platform.

In OZIX, I/O operations can be targeted to mass-storage devices through any of the following interfaces:

- The Application Programming Interface (API)
- The Dataspace Interface
- The Container Interface
- The Character Device Interface
- The Block Device Interface

Figure 21 shows various I/O interfaces that are available through the mass-storage subsystems in OZIX.

Figure 21: OZIX Mass-storage I/O Interfaces



The API and dataspace interfaces are discussed in the the OSF Application Programming Interface Functional Specification, and the Fault-recoverable Dataspace Subsystem Functional Specification. As illustrated in Figure 21, the container subsystems, together with the device drivers and hardware provide support for the ABA container interface and both block and character device interfaces.

8.4.1 Mass Storage Components

Supporting a system that processes large numbers of transactions imposes several stringent requirements on the mass storage. The primary requirements are:

- Seven-days-a-week, twenty-four-hours-a-day (7x24) operation, while handling terabytes of data
- Very high I/O rates that facilitate transaction turnaround time of 2 seconds or less
- Very high data availability

As illustrated in the bottom portion of Figure 19, OZIX mass storage consists of two major components:

- Containers
- Device drivers

8.4.1.1 Structuring Containers

As discussed in Section 8.2, the dataspace layer of ABA views mass storage in terms of containers, which are ABA abstractions that represent physical devices. Each container is identified by a unique global name, a UID, and a set of properties. Container properties represent state information in the same manner as the EMA-defined attributes.

A single physical unit is represented as a base container. In OZIX V1, base containers only represent random access devices, such as magnetic disks; it is undetermined whether sequential access devices, such as magnetic tapes, are to be represented by containers in later releases.

Containers can be combined to form *compound containers*, which present multiple base containers as a single logical device to the dataspaces layer. For example, disk shadowing is implemented by combining two base containers, which are logical representations of, say, two magnetic disk devices, into a single shadowed compound container. To the dataspace layer, the shadowed container is viewed as offering higher availability than either of the two base containers from which it is built. The OZIX I/O strategy is to offer a range of features based on compound containers. The following compound containers are expected to be available at V1 FRS:

- Shadowed container—This container consists of two or more constituent containers. Data is replicated in each of the constituents. Replication of data increases data availability and enhances data reliability.
- Striping container—This container consists of two or more containers on which data is distributed in a manner that provides optimum throughput and low latency. Applications that need high data transfer rates can use this container to meet their transfer rate requirements.
- Linear catenation—This consists of two or more containers to form a large virtual device. The address spaces are simply catenated. This compound container is suitable for holding large files that do not require high transfer rates.

The following compound containers are expected after V1 FRS:

- Caching container—This is made up of two containers: one is a large, relatively slow backing store device (for example, a nine inch disk); the other is a relatively small, very fast device (for example, a solid state device). The fast device is used as a cache.
- Logging container—This consists of three containers: a store, a backup copy, and a log. Updates are directed to both the store and the log. Reads are satisfied from the store. The store can be rebuilt from the log and the backup copy.

It is sometimes the case that all of the storage media represented by a container cannot be fully utilized by a single client, such as a file system or a database application. For this reason, containers can be sub-divided into *subcontainers*. Subcontainers offer the advantage of better media utilization, while retaining the properties of the container. For example, two relatively small file systems, both requiring high availability and reliability features can be created by sub-dividing a shadowed compound container.

From the above discussion on base, compound and sub-divided containers, it is clear that container abstractions can be realized in a general way. For example, several base containers can be combined to form a compound container, which can then be sub-divided to form several subcontainers. The lower portion of Figure 19 illustrates a few ways containers could be structured.

8.4.1.2 Device Drivers

A device driver is a collection of routines and data structures used by the OZIX system to process an I/O request. OZIX mass storage supports access to DSA (both DSA-1 and DSA-2) devices, as well as SCSI devices. Devices connected to an industry standard bus, such as Future Bus, may also be accessible in the future. The types of mass storage devices that are accessible are as follows:

- Magnetic disks
- Magnetic tapes
- Solid state devices
- Optical disks

- CDROM

OZIX users can directly access mass storage device drivers through either the *block device* interface or the *character device* interface. For block oriented devices like disks and tapes, a character device interface is quite unstructured and is therefore sometimes termed a *raw* interface.

DSA storage devices are accessible via the Computer Interconnect (CI) network, using the Storage Communication Architecture (SCA). SCA is divided into four distinct layers:

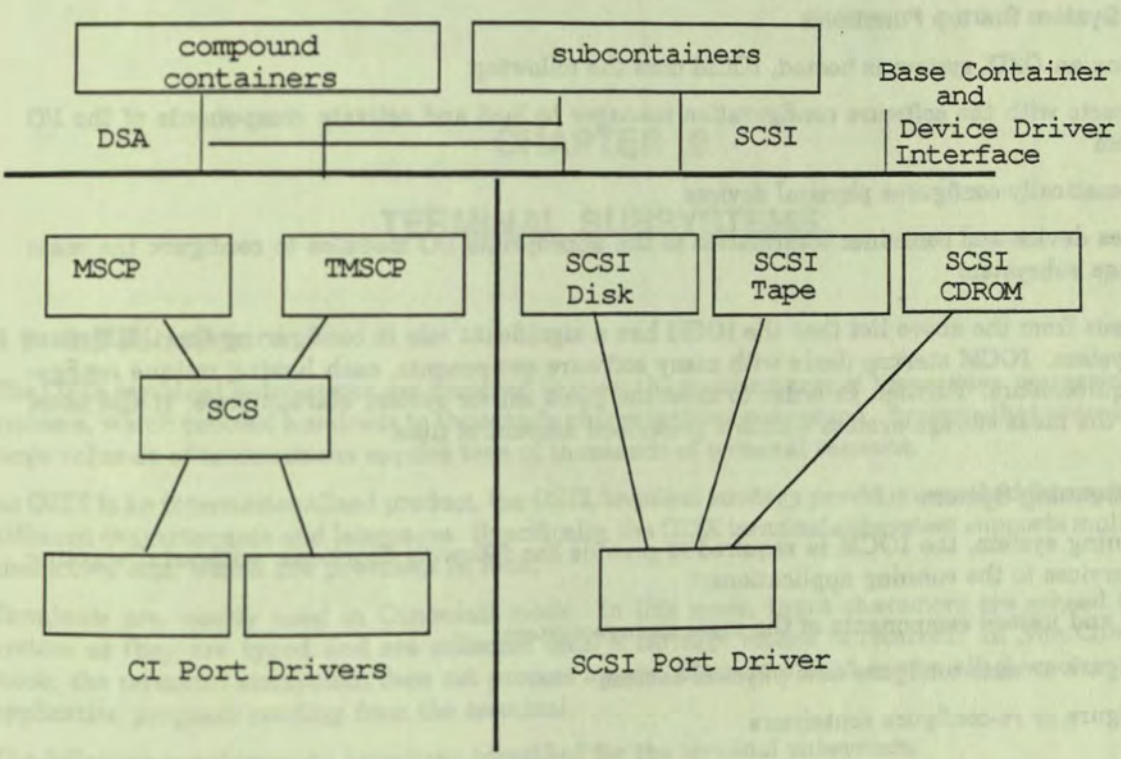
- Layer 0 is the Physical Interconnect layer
- Layer 1 is the Port/Port Driver (PPD) layer
- Layer 2 is the System Communication Services (SCS) layer
- Layer 3 is the System Application (SYSAP) layer

DSA device drivers are SYSAPs that use SCS to communicate with their counterparts residing in device controllers. Random access DSA devices are accessed through a mass storage control protocol (MSCP) class driver. Tape devices are accessed through a tape mass storage control protocol (TMSCP) class driver.

The system communication services (SCS) layer is responsible for managing connections between communicating SYSAPs. SCS performs some of the functions of the transport layer (layer 4), as well as some of the functions of the session layer (layer 5) of the OSI reference model.

The PPD layer is implemented as a combination of hardware and software. This layer performs some of the functions of the datalink layer (layer 2), the network layer (layer 3), and some of the functions of the transport layer (layer 4) of the OSI reference model.

Figure 22: Device Driver Components



8.4.2 Security

Mass storage components are required to handle data belonging to multiple users, as well as data belonging at multiple security levels. Thus, mass storage components must be defined as part of the trusted computing base (TCB). Being a part of the TCB imposes stringent requirements on the code. The mass storage components must satisfy the following requirements:

- Referring to objects—All objects must be accessed by way of the reference monitor
- Reusing objects— Any object that is reused must be (virtually or actually) zeroed
- Processing requests—All mass storage code must take the necessary precautions (for example, copy in a protected area) before processing a request

8.4.3 Management

OZIX I/O configuration manager (IOCM) provides the mechanisms to manage the hardware and software components of the OZIX mass storage system. The installed base of an OZIX system is expected to contain thousands of mass storage devices, all of which are to be monitored and controlled by the IOCM. The IOCM provides the following:

- A management interface that conforms to OSG Enterprise Management Framework
- System startup functions
- Adaptive management of I/O components on a running system

- System shutdown and restart functions

8.4.3.1 System Startup Functions

At the time an OZIX system is booted, IOCM does the following:

- Interacts with the software configuration manager to load and activate components of the I/O system
- Automatically configures physical devices
- Passes device and container information to the appropriate I/O modules to configure the mass storage subsystem

It is obvious from the above list that the IOCM has a significant role in configuring the OZIX mass storage system. IOCM startup deals with many software components, each having unique configuration requirements. Further, in order to meet the goals set for system startup time, IOCM must configure the mass storage system within a prescribed amount of time.

8.4.3.2 Running System

On a running system, the IOCM is required to provide the following functions, without disrupting regular services to the running applications:

- Load and unload components of the mass storage system
- Configure or auto-configure new physical devices
- Configure or re-configure containers

The IOCM can be queried for information regarding the states of the mass storage components.

CHAPTER 9

TERMINAL SUBSYSTEMS

9.1 Introduction

The OZIX terminal subsystems are designed to meet the requirements of transaction processing (TP) systems, which process hundreds to thousands of transactions per second. Systems that process such large volumes of transactions require tens of thousands of terminal sessions.

As OZIX is an internationalized product, the OZIX terminal strategy provides a method for supporting different character sets and languages. Specifically, the OZIX terminal subsystem supports multi-byte characters sets, which are prevalent in Asia.

Terminals are mostly used in *Canonical mode*. In this mode, input characters are echoed by the system as they are typed and are collected until a carriage return is received. In *Non-Canonical mode*, the terminal subsystem does not process special characters, and passes all characters to the application program reading from the terminal.

The following requirements have been identified for the terminal subsystem:

- Conformance to OSF and POSIX standards for interactions with terminals
- Support multiple character sets
- Support diverse terminal connection mechanism
- Conform to OZIX system management
- Conform to OZIX security model

9.1.1 Goals

The terminal subsystem has the following goals:

- Conform to OSF and POSIX standards for terminals.

  This goal is realized by supporting all of the terminal attributes defined by POSIX, including those for process control and timed input.

- Internationalization support.

  The terminal subsystem will support input and output of multi-octet characters which are needed to support many of the world's languages. In addition, individual character-set/language modules may handle the needs of particular languages. For instance, a method of converting phonetic input into Japanese Kanji characters will be supplied as a "character-set specific" module that plugs into the OZIX terminal driver

- Support for different connection methods.

Terminals may be connected to an OZIX system in a number of different ways. It should be possible to support a new connection method without having to rewrite the terminal subsystem. The primary method of connecting terminals to an OZIX system will be through the use of terminal servers which run network protocols such as LAT or TELNET.

- Scalability.

  The terminal subsystem is designed to support tens of thousands of terminal sessions. To support this, there will be a method of creating terminal connections "on the fly" without the need for a preconfigured terminal database.

- Provide for a robust I/O configuration management.

  This goal is realized by conforming to the EMA specifications. The terminal subsystem has a management interface, as well as descriptions for all terminal subsystem objects and object attributes.

- Provide support for the line disciplines specified in POSIX

The following are non-goals for version 1.0 OZIX terminal subsystem:

- Direct support for FORMS
- Provide support for real time I/O
- Support line disciplines other than that specified in POSIX
- Support modems

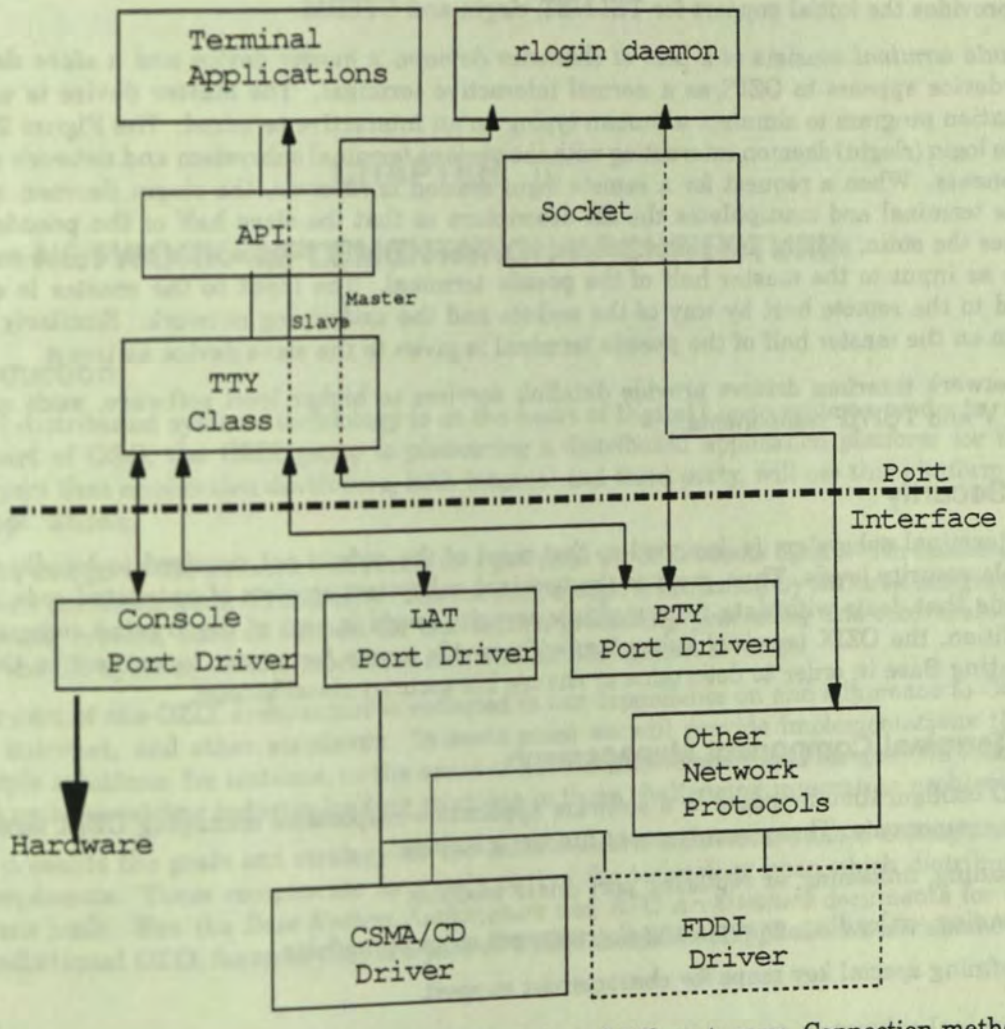
9.2 Terminal Components

OZIX terminal subsystem is divided into two major modules:

- Terminal Class Driver—This driver is responsible for providing user sessions as well as functions for data presentation (thus representing the session layer and the presentation layer of the OSI reference model). In OZIX V1, the terminal subsystem provides only the TTY class driver, which follows presentation rules specified by POSIX standard 1003.1.
- Terminal Port Driver—This driver is responsible for moving uninterpreted data between a hardware interface or network software module and the terminal class driver.

Figure 23 shows various components of the terminal subsystem along with the network drivers.

Figure 23: Terminal Subsystem and Network Drivers



Terminals may be connected to an OZIX system in a number of different ways. Connection methods include:

- Console terminals
- LAT terminals
- Network terminals (TELNET, rlogin, CTERM)

A Console terminal port driver interacts with the hardware console subsystem to make console terminal a path through which a system manager, maintenance engineer, or system user can communicate with the OZIX system.

OZIX provides local area networks to facilitate high speed communications within a relatively short distance (for example, a campus). OZIX local area network uses a physical interconnect, which is a single, shared network channel. This physical interconnect is used by network interface drivers such as the Ethernet driver and the FDDI driver.

The LAT provides functions similar to the OSI reference model network layer, transport layer and session layer. The LAT uses the network interface drivers to provide LAT terminal ports.

OZIX provides a pseudo terminal (PTY) port driver to simulate the actions of terminals in software. PTY provides the initial support for TELNET, rlogin and CTERM.

A *pseudo terminal* consists of a pair of character devices: a *master* device and a *slave* device. The slave device appears to OZIX as a normal interactive terminal. The master device is used by an application program to simulate a human typing on an interactive terminal. The Figure 23 shows a remote login (rlogin) daemon interacting with the various terminal subsystem and network subsystem components. When a request for a remote login session is received, the rlogin daemon allocates a pseudo terminal and manipulates the file descriptors so that the slave half of the pseudo terminal becomes the stdin, stdout, and stderr for a login process. The PTY driver funnels writes on the slave device as input to the master half of the pseudo terminal. The input to the master is eventually carried to the remote host by way of the sockets and the underlying network. Similarly, anything written on the master half of the pseudo terminal is given to the slave device as input.

The network interface drivers provide datalink services to higher level software, such as DECnet Phase V and TCP/IP components.

9.3 Security

OZIX terminal subsystem is designed so that most of the code is not required to handle data from multiple security levels. Thus, most of the terminal subsystem consists of untrusted code. However, any code that deals with data from multiple security levels is part of the trusted computing base. In addition, the OZIX terminal subsystem will provide a way for a user to connect to the Trusted Computing Base in order to determine or change his security classification.

9.4 Terminal Component Management

The I/O configuration manager is a software application responsible managing OZIX terminal subsystem components. These management functions include:

- Loading, unloading, or replacing port driver modules
- Loading, unloading, or replacing character-set support modules
- Defining special key maps for character-set support

The terminal subsystem co-operates with the IOCM to meet the system goals of 7X24 operation and system recovery time.

CHAPTER 10

NETWORK IMPLEMENTATION ARCHITECTURE

10.1 Introduction

Networks and distributed systems technology is at the heart of Digital's open systems products. As an integral part of OSG, the OZIX group is pioneering a distributed application platform for the 1990s. We expect that application developers, both internal and third party, will use this platform to build their applications.

The underlying design of the network will result in a powerful set of products upon which customers can rely for high performance and reliability. The network design is enhanced by features integrated into the operating system, such as support for transaction processing, journaling and recoverability primitives in the file system, remote procedure call, and authentication services.

An important part of the OZIX architecture is reflected in our dependence on and adherence to OSF, POSIX, ISO, Internet, and other standards. In some cases we will provide implementations that support multiple solutions, for instance, in the areas of authentication and name service. We look to NaC to assist us in providing industry-leading solutions to these challenging integration problems.

This chapter presents the goals and strategy for the OZIX network architecture and a description of the major components. These components form only part of the technology upon which distributed applications are built. See the *Base System Architecture* and *RPC Architecture* documents for descriptions of additional OZIX features that are part of OZIX's distributed applications environment.

10.2 Goals

The goals of the OSG network strategy are to:

- Exploit distributed systems and networks for Digital's advantage in the open systems market
- Support one family of open systems products in a network
- Provide a distributed computing environment using *internet* and OSI standards
- Provide DECnet/OSI distributed computing integration
- Lead the migration of open systems to OSI standards
- Propose and drive open systems standards for distributed systems
- Provide connectivity with VMS systems without being VMS-dependent

10.3 Strategy

OSG products, which include workstations and servers, share a common network and distributed systems architecture. In accordance with this architecture, OZIX will implement the network components and distributed services for the server environment.

- Digital's open systems products will implement this distributed systems architecture using TCP/IP-based extended local area network (LAN) subnets. Wide area network access will be provided through a DECnet/OSI backbone using the Internet Portal product.
- OZIX systems will provide *Phase V DECnet/OSI* end-node support, making them accessible to Phase V-compliant systems anywhere on the extended Phase V network.
- Interoperation between VMS and internet protocols on the subnet will be available through the *VMS/ULTRIX Connection*, a VMS layered product. Communication with VMS systems not on the subnet will be available through a DECnet/OSI backbone using the Internet Portal product.
- The *network application programming interface* (API) presented to applications will be common across internet transport protocols, ISO transport, and DECnet/OSI Session Control for OZIX and ULTRIX. It is not a goal of the API to hide semantic differences among the transport protocols.
- OZIX systems will provide a network management interface that is integrated with system management. Network management of DECnet/OSI and TCP/IP will be integrated at the user interface. The network management programming interface is common on OZIX and ULTRIX.
- Digital's open systems will adhere to OSF standards. Where deficiencies are noted, Digital will propose its distributed systems architecture components. Digital will lead the market by integrating these services.
- Workstation services include:
  - Name services using *Distributed Name Service* (*DNS*) and *Berkeley Internet Name Domain Service* (*BIND*)
  - Time services using *Digital Time Synchronization Services* (*DTSS*) and *Network Time Protocol* (*NTP*)
  - File services using *Network File System* (*NFS*)
  - Wide area network access through DECnet/OSI routers and x.25 gateways
  - Authentication services using *Kerberos* and *Digital Authentication Security Service* (*DASS*)
  - Remote procedure call services using *DEC RPC*
- Digital will accelerate the movement of the market toward OSI by providing an OSF reference implementation that is compliant with OSI standards.
- Digital will add value in the distributed systems area by providing solutions in naming, network management, RPC, security, authentication, common application subsystem interfaces, and time services.
- Digital's open systems will provide x.25 access and SNA interoperation through low-cost gateways, consistent with the corporate strategy. If direct connection becomes important in certain markets, third-party solutions will provide this capability.
- PC integration for Digital's open systems will be provided by DECnet/OSI and third-party solutions.

10.4 Performance

The OZIX network will be co-designed with the operating system in order to fully utilize the advanced capabilities provided by OZIX. The network will provide state-of-the-art performance by exploiting the inherent parallelism of symmetric multi-processing (SMP) through the use of the OZIX thread architecture.

The effective use of any thread architecture requires that it be designed into the network implementation, since it affects the structure of the internal network database, influences the choice of an appropriate database locking granularity, and demands the studious avoidance of unnecessary thread context switching.

As a result of these considerations, the OZIX network will enjoy a significant performance advantage when compared to implementations that do not employ threads as a fundamental design element.

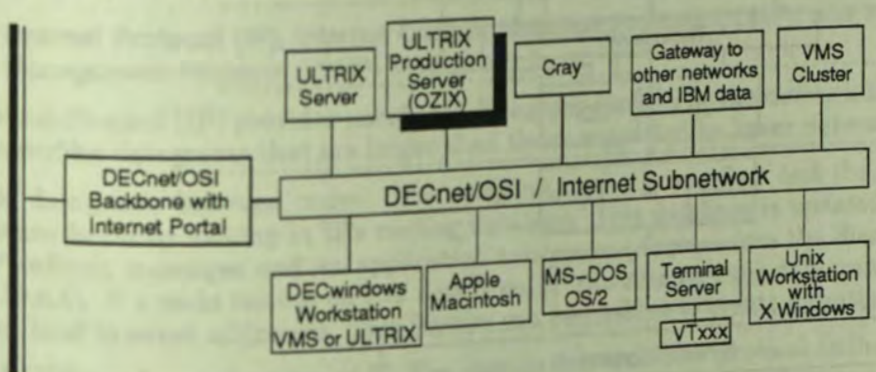
In order to ensure the performance advantage gained by the intrinsic parallelism of the OZIX network, careful attention is given to the fundamental issues applicable to any good network design. These include, for example, the use of buffer management mechanisms that avoid data copying wherever possible, as well as the extensive use of direct procedure calls between network layers.

10.5 Architectural Overview

The OZIX networking environment is designed to support heterogeneous systems (workstations and servers) participating in both local and wide area networks. The product is a combination of DECnet/OSI Phase V and internet components. Wide area network access is provided by a DECnet/OSI backbone gateway and an internet portal. TCP/IP is the protocol used for ULTRIX interoperation on the local area network; DECnet/OSI provides interoperability with VMS systems and other vendors' OSI end nodes (see Figure 24). The ultimate goal is to encourage the industry to move toward OSI, thus a single network solution. We look to NaC to provide leadership in influencing this migration to OSI.

OZIX strives to provide Internet functionality matching or exceeding any other competitor; it will meet all relevant IP standards defined at least 6 months prior to field test.

Figure 24: Network Architecture Overview



10.6 Internet Components

Six protocols form the basis for an internet network node (see Figure 25). Internet networks do not provide distinct protocols at ISO's session and presentation layers.

At the transport layer:

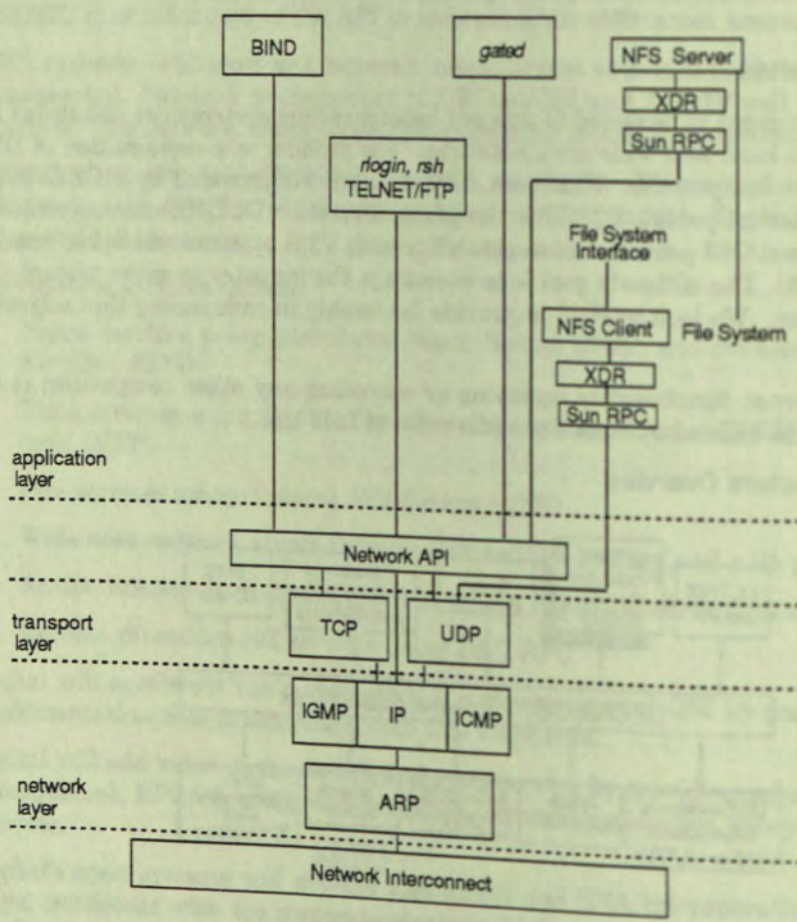
- *Transmission Control Protocol (TCP)*
- *User Datagram Protocol (UDP)*

At the network layer:

- *Internet Protocol (IP)*
- *Internet Control Message Protocol (ICMP)*
- *Internet Group Management Protocol (IGMP)*
- *Address Resolution Protocol (ARP)* (includes *Reverse Address Resolution Protocol (RARP)* and *Proxy ARP*)

The following sections detail each layer and its associated protocols.

Figure 25: Internet Network Components



10.6.1 Application Layer Code

Application code accesses OZIX network services through the network API which includes both Berkeley *sockets* and the *X/OPEN Transport Interface (XTI)*. Portable application code using either of these interfaces can be moved to OZIX.

The Berkeley Internet Name Domain (BIND) Service, an implementation of the Internet Domain Name Service, is ported from BSD4.3. BIND is a network service that allows client systems to obtain translations of host names into internet addresses.

It is expected that the Open Systems Components and Resources (OSCR) group will maintain the set of common application codes for both OZIX and ULTRIX/OSF.

10.6.2 Transmission Control Protocol (TCP) Module

At the transport layer, TCP provides a reliable, connection-oriented service that guarantees that messages are delivered to their destination in the order they were originally sent. It uses the network service of the Internet Protocol (IP) to communicate between peer TCP protocol layers.

10.6.3 User Datagram Protocol (UDP) Module

At the transport layer, UDP provides an unreliable, connectionless service commonly referred to as a *datagram* service. Applications using this service can send datagrams to any other UDP user, either local or remote. Because it is an unreliable service, datagrams may be lost, delivered out of order, or may be duplicated.

An outgoing datagram can be sent to a single UDP session on a distinct system, or it can be broadcast to many systems. The semantics of broadcast transmission, reception, and propagation are implemented in the IP network layer.

UDP datagrams can also be sent to and received from IP multicast groups. A multicast group is a special class of Internet addressing used by cooperating hosts to provide broadcast-like transmission and reception without the cost normally associated with broadcasting. The members of group are the only hosts that incur the cost of processing the multicast datagram.

10.6.4 Internet Protocol (IP), Internet Control Message Protocol (ICMP), and Internet Group Management Protocol (IGMP) Modules

The Internet Protocol (IP) provides unreliable datagram services at the network layer. IP fragments and reassembles datagrams that are larger than those supported by lower network layers.

IP moves datagrams between nodes. The outgoing network interface and the IP address of the next-hop are found by looking in IP's routing database. This database is updated by a combination of ICMP redirect messages and an application subsystem program (see the discussion of *gated* in Section 10.6.5). If a node resides on two networks, IP can also forward datagrams not destined for one of the local internet addresses. This feature can be enabled to create a routing node.

TCP and UDP are layered on top of IP. The destination transport protocol (either TCP or UDP) is selected by a protocol ID in the IP datagram. IP is subsequently layered on any subnet-dependent device that can support IP's concepts of addressing and datablocking. Device-dependent convergence layers provide the mapping of IP addresses to the subnet-dependent device's addressing scheme. OZIX supports the Address Resolution Protocol (ARP) (discussed in Section 10.6.6), which is responsible for mapping internet (logical) addresses to Ethernet and IEEE 802.3 (physical) addresses.

IP is not designed to provide reliable delivery. The Internet Control Message Protocol (ICMP) is used to provide feedback in the internet communications environment. Since ICMP is based on IP for its communications, an ICMP datagram has all the characteristics of a normal IP datagram, for example, it may not be delivered. ICMP is an integral part of IP and must be implemented by every IP module.

The Internet Group Management Protocol (IGMP) is used by IP hosts to communicate their multicast group memberships to any immediately neighboring multicast routers. Currently, multicast routing is experimental in the Internet, but the protocol is required for Level 2 IP multicast support. Due to their experimental classification, multicast routers are currently not supported on OZIX.

10.6.5 Routing Database Maintenance

The routing database management daemon, *gated*, is from Cornell University. It accesses IP through the network API and receives the External Gateway Protocol (EGP), Routing Information Protocol (RIP), Defense Communication Network's HELLO routing protocol, and Internet Control Message Protocol (ICMP) messages that affect the routing database. It interprets these messages and communicates directly with IP to change the internal copy of the routing database.

A new internal routing protocol, Open Shortest Path First (OSPF), is being introduced in the Internet community. OZIX will track its use and, if appropriate, provide an implementation of OSPF at FRS.

10.6.6 Address Resolution Protocol (ARP) and Proxy ARP Modules

At the network layer, Address Resolution Protocol (ARP) converts 32-bit internet addresses to 48-bit Ethernet and IEEE 802.3 addresses. Reverse ARP translates Ethernet and IEEE 802.3 addresses to internet addresses.

Proxy ARP allows a gateway to answer ARP requests on behalf of a destination node not on the same subnet.

10.7 DECnet/OSI Components

The components that make up DECnet/OSI on OZIX are presented according to their layer in the network model (see Figure 26).

At the application layer:

- *File Transfer and Access Management (FTAM)*
- *Virtual Terminal Protocol (VTP)*
- *DECnet/OSI Copy Program (DCP)*
- *dlogin*
- *Distributed Name Service (DNS) Clerk*
- *Digital Time Synchronization Services (DTSS) Client*
- x.400

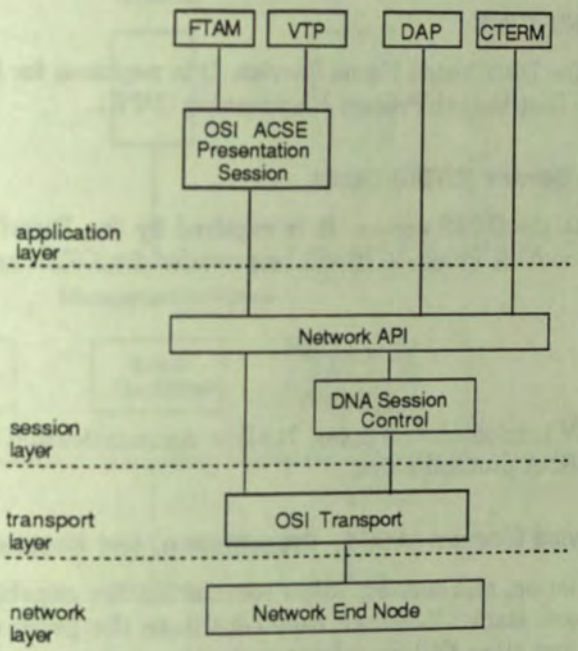
At the session layer:

- DNA Session
- OSI Session

At the transport layer:

- OSI Transport Protocol (TP4)

Figure 26: DECnet/OSI Network Components



10.7.1 Application Level Components

The application level components that OZIX provides are a mixture of DECnet-specified and OSI-specified components. DECnet-specified components will be ported from ULTRIX. OSI-specified components may not exist in a portable form and may have to be written or purchased from a third party.

10.7.1.1 File Transfer and Access Management (FTAM)

FTAM is the OSI method for file transfer and access over the network. The ULTRIX version of FTAM will be ported to OZIX.

10.7.1.2 Virtual Terminal Protocol (VTP)

VTP is the OSI virtual terminal protocol used to request outbound remote terminal sessions. VTP will be ported from ULTRIX.

10.7.1.3 DECnet/OSI Copy Program (DCP) and the File Access Listener (FAL) Daemon

DCP is the client portion of the DECnet/ULTRIX file transfer utility. The FAL daemon is the server portion, and responds to DCP requests. This facility is used to interchange files between OZIX systems or systems running DECnet/OSI Phase V.

10.7.1.4 dlogin and the dlogind daemon

dlogin is the DECnet virtual terminal program from DECnet/ULTRIX. It is used to request outbound remote terminal sessions. The *dlogind* daemon is used to accept inbound terminal sessions from remote DECnet/OSI systems. Together they implement the DECnet/OSI CTERM virtual terminal protocol. *dlogin* and *dlogind* will be ported from ULTRIX.

10.7.1.5 Distributed Name Service (DNS) Clerk

The DNS Clerk is the local interface to the Distributed Name Service. It is required for DECnet/OSI Phase V. It will be provided for OZIX by Distributed Process Engineering (DPE).

10.7.1.6 Digital Time Synchronization Service (DTSS) Client

The DTSS Client is the local interface to the DTSS server. It is required by the Distributed Name Server, which is itself required for DECnet/OSI Phase V. It will be provided for OZIX by DPE.

10.7.2 Session Layer Components

10.7.2.1 DNA Session

DNA Session is the DECnet/OSI Phase V session control layer. It allow communication capability to those applications that require the full DNA protocol stack.

10.7.2.2 OSI Association Control Service Element (ACSE), Presentation, and Session

The upper layers of OSI (ACSE, presentation, and session) allow communication capability between applications that use the full OSI protocol stack. Together, they constitute the preferred interface between DECnet/OSI Phase V systems and other OSI-compliant systems.

10.7.3 Transport Layer Component

DECnet/OSI supports ISO's *Transport Class 4 (TP4)*, which is the most robust of the ISO-specified transports. TP4 provides the capability to multiplex many concurrent network sessions over an unreliable physical link.

The ISO TP0 and TP2 transport classes, which support non-multiplexed (TP0) and multiplexed (TP2) communications on reliable, error-free physical links, will be considered for future releases of OZIX.

10.7.4 DNA Network Layer

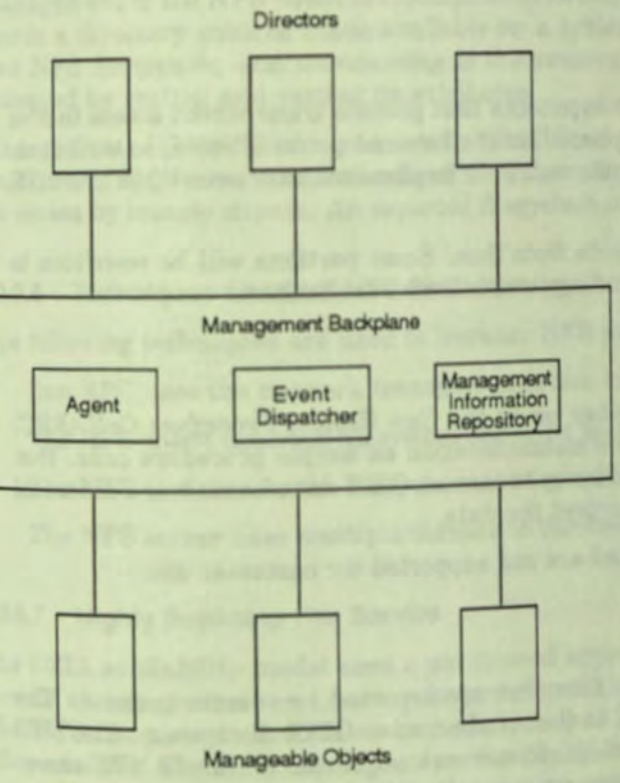
The network layer (end node routing) on OZIX conforms to the DECnet/OSI Phase V network layer architecture and supports DECnet/OSI Phase V-style addresses. The OZIX implementation of the network layer provides end node support only. Support for multiple physical links on Ethernet (multi-link) will be provided to improve throughput and availability.

10.8 Management Components

In OZIX, system and network management are integrated to present a seamless, coherent management picture. System and network components are managed together using the same *management backplane*. Local and remote objects can be managed using this backplane.

he backplane provides dynamically installable protocol towers for communication that conforms DNA and Internet specifications. Currently the protocols supported are *Common Management formation Protocol* (*CMIP*), *CMOT* (CMIP over TCP/IP), and *Simple Network Management Protocol* (*SNMP*).

igure 27: Network Management Components



0.8.1 The Backplane

n Figure 27, directors manage the system and *network objects* via the backplane services. The ackplane provides a common API to the directors for directives, and a common API to the *manageable bjects*.

0.8.2 The Director

The *director* is an application-level program that generates management requests. The interface s graphical in nature, although it derives from a textual-based system, *Network Control Language* (*NCL*). NCL will be extended and renamed *MCL* (*Management Control Language*) to incorporate both ystem and network management directives. Internet, DECnet, and system manageable objects will ll be manageable by this director.

0.8.3 The Agent

The *agent* is the intermediate recipient of all management directives. The agent locates and passes he request to the appropriate manageable object.

10.8.4 The Event Dispatcher

The *event dispatcher (EVD)* is a system component that accepts internal state change notification from any manageable object in the system. It dispatches the event to local and remote systems for further processing. The event dispatcher conforms to the DNA Phase V EVD functional specification, although its functionality is extended to encompass both system and network event postings.

10.9 Network File System

The *Network File System (NFS)* is a collection of components that provide transparent access to files located on remote systems. These components are based on the Internet protocol suite, as well as de facto standards that originated from Sun Microsystems. This implementation uses V2 of the NFS file protocol.

NFS is based on the V4 release of the reference code from Sun. Some portions will be rewritten to accommodate performance, B2 security, multithreading, and internationalization.

10.9.1 Sun Remote Procedure Call

The NFS components communicate with one another using the *Sun Remote Procedure Call (RPC)* mechanism. This allows programs to model remote communication as simple procedure calls. Sun RPC uses the *External Data Representation (XDR)* library to convert OZIX data forms into a canonical form so that any implementation can properly interpret the data.

Sun RPC and XDR are provided for use by NFS and are not supported for customer use.

10.9.2 NFS Client Subsystem

The NFS client gives OZIX applications access to files that are exported by remote systems. The application interface to the NFS client is identical to that of the native OZIX file system. The NFS client converts OZIX file I/O requests for remote files into network requests. A remote NFS server performs the actual file accesses on behalf of the requesting application.

The NFS Client communicates with the User Datagram Protocol (UDP) subsystem using the network transport interface instead of using the network Application Programming Interface (API).

10.9.3 NFS Server Subsystem

The NFS file server subsystem accepts client requests from remote nodes and executes them as local file system requests. The server will implement the duplicate request filtering enhancements first introduced with ULTRIX.

Two other subsystems assist the file server. The port mapping server allows callers to properly locate the server; the filesystem mounting server distributes initial access to exported filesystems.

10.9.4 NFS Locking Server Subsystem

The NFS locking subsystem allows clients on remote nodes to lock files on the system where the locking server is running. The locking server works with the status monitor subsystem, which is responsible for connection management issues.

From a client's perspective, if a lock is requested on a local file, OZIX performs the requested action directly. If the lock is on an NFS file being served by a remote system, a locking request is sent to the locking server on the remote system.

10.9.5 NFS Management

NFS management is integrated with the OZIX system management framework. Generic management of each of the NFS components consists of starting the component, stopping the component, getting attributes and setting attributes.

Management of the NFS client is accomplished through management of "filesystem objects." A filesystem is a directory subtree made available on a system through NFS. *Mounting* is the creation of a new NFS filesystem, and *unmounting* is the removal of an NFS filesystem. An NFS filesystem is managed by getting and setting its attributes.

Management of the NFS server is accomplished through management of "exported filesystem objects." An exported filesystem is an entire local filesystem, directories, or a single file which is made available for access by remote clients. An exported filesystem is managed by getting and setting its attributes.

10.9.6 Techniques used for NFS Performance

The following techniques are used to increase NFS performance.

- Sun RPC uses the network transport interface to UDP which minimizes copying of data.
- The NFS client uses the filesystem buffering mechanisms for read ahead and write behind.
- The NFS server uses the filesystem caching mechanisms to avoid unnecessary disk accesses.
- The NFS server uses multiple threads to increase concurrency.

10.9.7 Highly Available File Service

The OZIX availability model uses a partitioned approach wherein a subset of the resources provide service at any given time. A secondary server is designated to recover state and resume service. The NFS components are stateless in that there is no required state to be recovered upon restart after a failure. The NFS components work with other highly available OZIX components such as the network transport, file system, and mass storage subsystems.

10.10 DEC Remote Procedure Call

A remote procedure call capability will be provided by *DEC RPC* V1.0. The ability to write applications using remote procedure calls facilitates development of distributed applications by allowing programs on various nodes across a local area network to register as providers of services, or *servers*. Consumers of these services, or *clients*, can then make requests of any server that has been registered as a provider of the desired service.

DEC RPC V1.0, which is based on Apollo's *Network Computer System* (*NCS*) version 1.5, is the first product offering defined by the joint DEC/Apollo RPC Architecture and will be bundled with ULTRIX V4.0 as well as with V1.4 of the VMS/ULTRIX Connection (UCX).

DEC and Apollo are jointly submitting RPC V2.0, an enhanced version of DEC RPC V1.0, to OSF in response to the Distributed Application Environment (DAE) Request for Technology (RFT). It is expected that RPC V2.0 will be the RPC chosen by OSF, although because of time constraints, OSF will likely include NCS V1.5 in OSF/1.

By providing DEC RPC V1.0, OZIX will be able to interoperate with ULTRIX and Apollo systems. If OSF accepts the joint DEC/Apollo RPC V2.0 submission, as it is fully expected to do, OZIX's level of interoperability will be greatly increased.

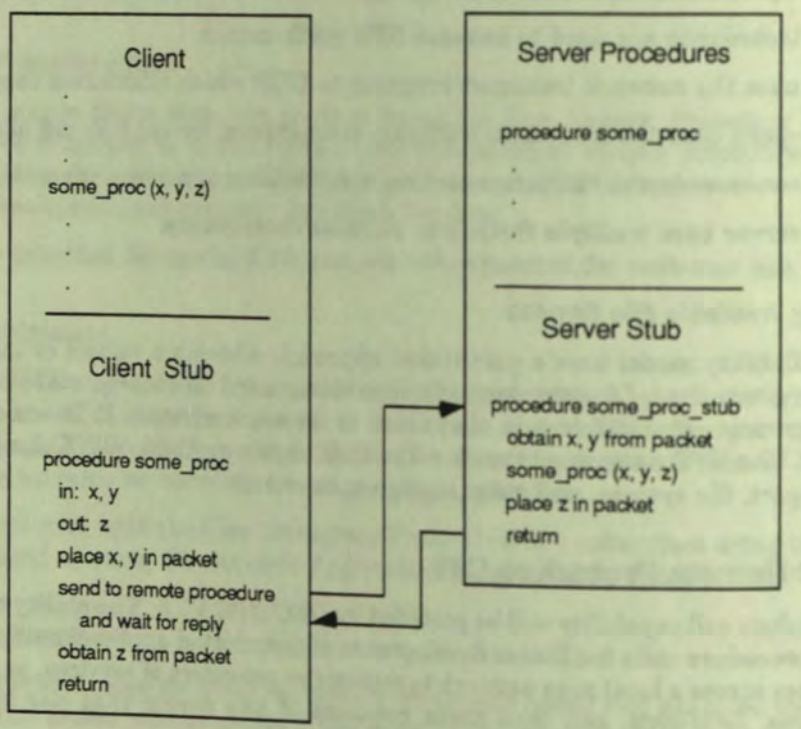
DEC RPC V1.0 uses the Berkeley socket interface to the UDP transport layer. It includes three major components: the *Network Interface Definition Language* (*NIDL*) Compiler, the RPC runtime library, and the *Location Broker*.

10.10.1 Network Interface Definition Language (NIDL) Compiler

The NIDL Compiler is the stub generator for DEC RPC. An NIDL interface definition rigorously describes the interface to a remote procedure: it defines the functions, procedures, and data types of the parameters that make up a remote interface.

Given an interface definition written in the Network Interface Definition Language, the NIDL Compiler produces C header files, and client and server stubs with the appropriate RPC runtime library calls. Figure 28 illustrates the role of client and server stub code in a remote procedure call.

Figure 28: Client and Server Stub Interaction



10.10.2 The RPC Runtime Library

The RPC runtime library actually implements the RPC functions. It is composed of routines that affect bindings, register and unregister remote interfaces, marshall parameters, provide server information, and access the network. Many of these routines are never directly called by clients or servers, but are instead called from the stubs that are generated by the NIDL Compiler.

10.10.3 The Location Broker

A server database is maintained by the DEC RPC Location Broker, allowing it to associate incoming client requests with an appropriate server. This mechanism requires that servers register information such as socket addresses and exported procedure definitions with the Location Broker.

It is expected that in future releases, DEC RPC will move towards using the Distributed Name Service (DNS), but in the initial release, it will use the Location Broker.

10.11 Network Interconnects

Initially OZIX will support only Ethernet and IEEE 802.3 as communications links. This is consistent with our strategy to provide wide area network access through Digital's OSI backbone gateways and IP portal products. OZIX's DECnet/OSI product will support only standard IEEE 802.3 and accommodate multiple controllers. Internet will use both IEEE 802.3 and Ethernet, but considers them logically separate network interconnects. Internet will also support multiple controllers.

As fiber distributed data interface (FDDI) products are made available to the hardware platforms supported by OZIX, FDDI will be incorporated as a network interconnect for both Internet and DECnet/OSI.

Direct access to the network interconnects is provided via two interfaces: the Data Link Interface (DLI) and the Packet Filter.

The DLI is presented as a distinct address family in the BSD socket portion of the network API. DLI endpoints are not available through the XTI portion. The appropriate access controls are used to prevent unauthorized use of the devices represented by the DLI. The functionality provided by an OZIX DLI endpoint is identical to that of ULTRIX DLI.

The Packet Filter is similar in function to the DLI, but provides a richer filtering mechanism[1]. The Packet Filter allows users to specify multiple, non-contiguous fields in a packet as the selection criteria. This is compared to the DLI wherein only one field is used for demultiplexing. The Packet Filter was designed to allow prototyping of network code in user mode while providing a relatively low overhead interface to the network.

10.12 IBM Interconnect Components

The components of the OZIX IBM Interconnect will make use of the planned DECnet/OSI Phase V SNA gateway products. The components fall into two categories; application components that can be used generically over OZIX or ULTRIX, taking advantage of the common API Libraries; and those that must be ported from ULTRIX- or VMS-based versions.

10.12.1 Generic IBM Interconnect Components

Applications interacting with the SNA gateway do so via DECnet/OSI. Hence, the portable applications and libraries already planned by the SNA Interconnect Group will be able to use the XTI and/or socket interfaces to DECnet/OSI/OZIX without modification. These products include:

- DTF File Transfer - Allows file transfer between hosts on connected DECnet/OSI and SNA Networks.
- DW 3270 Terminal Emulation - Provides emulation of IBM's 3270-style terminals on DEC terminals and DECwindows.

[1] See "The Packet Filter: An Efficient Mechanism for User-level Network Code", Jeffery C. Mogul, Richard R. Rashid, and [illegible] Austin, Texas, November 1987.

- 3270 Data Stream - Provides a programming interface for our users to interact with SNA host-based applications designed to be accessed via 3270 style terminals.
- SNA Application Programming Interface - Provides a programming interface for our users to interact with SNA using LU0 protocol.
- APPC/LU6.2 - Provides a programming interface for our users to interact with SNA using LU6.2 protocol.

10.12.2 Ported IBM Interconnect Components

Most components that may require some conversion effort include managing the SNA gateway products.

SNA gateway management consists of downline-loading an image for a given SNA gateway hardware product, setting, reading, and changing its manageable objects once it is running, and monitoring its operation.

10.12.2.1 Booting the SNA Gateway

An SNA gateway hardware product will be booted via the *Maintenance Operations Protocol (MOP)*, already planned as a part of the DECnet/OSI product for OZIX. No additional work will be required.

10.12.2.2 SNA Gateway Management

The manageable entities within an SNA gateway product will have to be understood by our network management utilities. Access to these manageable entities will be via the CMIP protocol, already planned for OZIX as part of the DECnet/OSI network management implementation. Definitions of the manageable entities will have to be loaded, but the definitions are expected to be independent of the operating system.

10.12.2.3 SNA Gateway Monitoring

Two monitoring applications are required and will have to be ported from other operating systems. These are the Common Trace Facility (CTF), which includes the capability to trace SNA gateway line activity, and SNAP, a gateway monitoring tool.

10.12.3 Access to OZIX via SNA

Products planned by Digital to facilitate access to data and applications from the SNA side of the gateway use existing protocols already planned for inclusion in OZIX. These products and their associated protocols are:

- Medusa (LAT) - Allows 3270 style terminals access to either IBM resources, or LAT access to DEC resources via Ethernet.

10.13 Application Programming Interface (API)

Two application programming interfaces (APIs) are provided to all OZIX-supported transport layers. The first form was developed at Berkeley for the BSD 4.x operating system and is called *sockets*. The second form was developed at AT&T for the System Vr3 operating system and is called the Transport Layer Interface (TLI). The TLI has been adopted and slightly changed by X/OPEN; it is now called X/OPEN Transport Interface (XTI).

OZIX provides both of these interfaces as part of the OSF API. The socket/XTI API is designed to be a very thin veneer over the network transport interface so as to not impact performance of the applications. All OZIX transport layers implement the network transport interface, which is the union of socket and XTI functions.

While there is a common API across DECnet/OSI and internet, the specification of the transport to be used is part of the call to the API. The issue of making this transparent to the caller is a potential area for OSG to contribute added value.

10.14 Network Security

10.14.1 Philosophy

The goal for OZIX to be both secure and interoperable presents an interesting challenge since, at this time, there are no widely accepted intravendor protocols that provide a reasonable level of security. Therefore, OZIX will take a phased approach to network security. In the first release, only a single network operating at a single security label will be provided.

Over a period of several releases, and as some of the expected security protocols become standardized, OZIX will provide more extensive support for multilevel network capabilities.

10.14.2 The Trusted Computing Base (TCB) Boundary

The networking software will not be trusted in the first release of OZIX. However, subsequent releases will require that some or all of the network software will be a part of the TCB. The network layer and at least some of the transport layer will be required to handle data from multiple layers simultaneously. Any cryptographic routines will have to be trusted, as these routines will be moving data between two security labels. Some authentication routines will also have to be trusted.

10.14.3 Performance Options and Tradeoffs

Many options are available to the networking software to allow tradeoffs between security, performance, and functionality. Some of these tradeoffs are listed below.

- Including more code in the TCB will reduce the number of trusted/untrusted transitions, resulting in increased performance. Not all network code needs to be in the TCB, but if any untrusted code resides in a performance-critical path between two trusted portions of the network, then it may be included in the TCB to improve efficiency.
- The specifications for the first release of OZIX networking are extremely restrictive. This is required to pass the government certification requirements. However, not all customers require this level of security. As an option, the security label at which the network runs could become a range of security labels, with the understanding that any data going over the network is likely to lose its exact label, and have a new one assigned by the receiving system.
- Additionally, authentication requirements could be relaxed for some customers, so that the user-name/password or soft proxy information sent by existing protocols could be trusted, giving access to a wider range of target user accounts.

10.15 Services

10.15.1 Authentication Services

OZIX will implement kerberos as its authentication mechanism. This will be achieved by modifying NFS to authenticate clients and servers using kerberos.

Kerberos currently resides in public domain and is being implemented by both Apollo and IBM. It has also been presented to OSF. While kerberos will serve as the initial authentication mechanism for OZIX, NaC's *Distributed System Security Architecture* (*DSSA*) will need to be integrated into OZIX. How this will be accomplished has not yet been determined.

10.15.2 Name/Directory Services

Several name services have been accepted, proposed, or required to support the protocols and services which are part of the OZIX strategy. These include the Internet Domain Name Service, Distributed Name Service (DNS), x.500, and the location broker for NCS.

OZIX and ULTRIX will support the integrated DNS/*hesiod*/BIND name server being developed by Distributed Process Engineering(DPE). This name server supports DNS semantics (as required by DECnet) and a BIND interface (as required by Internet), thereby supporting both networks. *Hesiod*, developed by project Athena at MIT, is an enhanced user interface to BIND. These, in combination with the back end update characteristics of DNS, will provide us with a superior name server. This name server will be presented to OSF by NaC.

10.15.3 Mail Services

Mail Services will be provided by Open Software Components and Resources (OSCR), the common software group within OSG. They are developing an x.400-based product that uses DNS and x.500 directory services. In addition, it will support *Simple Mail Transport Protocol* (*SMTP*) and *Message Router* (*MR*) interfaces. This product will be common for both OZIX and ULTRIX.

10.15.4 Boot Services

Ozix will support Digital's Maintenance Operations Protocol (MOP). The internet booting protocol, *BOOTP*, will be supported, although the method of support is a marketing issue to be resolved at the appropriate time. BOOTP is a proposed standard documented in RFCs 951 and 1084. BOOTP thus far has not been adopted by other major workstation competitors such as Apollo.

An alternative strategy being investigated would be to support the extensions to RARP as developed by SUN.[2] These extensions are commonly known as *BOOTPARAM*.

10.16 Network Architecture Components for OZIX

A comprehensive list of the OZIX networking architecture components is shown below. Asterisks mark those items that are still being studied for inclusion in OZIX.

[2] SUN is a trademark of Sun Microsystems Inc.

Communication Services

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- Internet Group Management Protocol (IGMP)
- External Gateway Protocol (EGP)
- Routing Information Protocol (RIP)
- Address Resolution Protocol (ARP)
- TP4 (ISO transport)
- ES-IS (ISO end node)
- UNIX domain sockets
- *Open Shortest Path First (OSPF) routing protocol

Communication Links

- Ethernet/IEEE 802.3

IBM Interconnect Components

Generic Components:

- DTF File Transfer
- DW 3270 Terminal Emulation
- 3270 Data Stream
- SNA Application Programming Interface
- APPC/LU6.2

Ported Components:

- Maintenance Operations Protocol (MOP)
- Common Trace Facility (CTF)
- SNAP - System Network Architecture Protocol (protocol analysis tool for SNA)

Access to OZIX via SNA:

- Medusa (LAT)

Application Programming Interface (API)

- X/OPEN Transport Interface (XTI)
- BSD sockets
- Network Computer System Remote Procedure Call (NCS RPC)
- Sun RPC (provided for Network File System interoperability only)

- DNA Session, OSI Session

File Store Services

- ftp (file transfer program)
- tftp (trivial file transfer program)
- rcp (remote copy program)
- Network File System (NFS)
- File Transfer and Access Management (FTAM)
- DECnet/OSI Copy Program (dcp)

Mail Services

- Simple Mail Transfer Protocol (SMTP)
- x.400 messaging

Name/Directory Services

- BIND/hesiod
- Distributed Name Service (DNS)
- x.500
- location broker for DEC RPC

Terminal Services

- Local Area Transport (LAT)
- TELNET
- dlogin, rlogin (remote login), rsh (remote shell)
- cterm
- OSI Virtual Terminal Protocol (VTP))

Network Management

- Common Management Information Protocol (CMIP), Common Management Information Service (CMIS), Network Command Language (NCL)
- Simple Network Management Protocol (SNMP)
- Internet Engineering Task Force (IETF) CMIP as recommended

Time Services

- DTSS - Digital Time Synchronization Services (DTSS)
- NTP - Network Time Protocol for internet

Boot Services

- Maintenance Operations Protocol (MOP) - Digital-proprietary
- BOOTP
- *BOOTPARAM

Security Services

- Kerberos
- Digital Authentication Security Service (DASS)
- Sun Secure RPC (for NFS interoperability)

CHAPTER 11

SYSTEM INSTALLATION AND SYSTEM DIRECTORY LAYOUT

11.1 Introduction

This chapter provides an overview to OZIX Version 1.0 system directory structure, system software installation, and layered application software installation. These subjects are covered in the same document because of their required integration to meet the goals of OZIX installations. This chapter first introduces the goals of OZIX software installation and then covers the system directory structure used to meet those goals. The discussion continues with a description of the system software installation and update procedure and concludes with layered application software installations and updates.

This chapter assumes the reader is familar with OZIX System Administration as described in the *OZIX System Administration Overview* [1]. Additionally, familiarity with the OZIX device and storage design, using the Attribute Based Allocation (ABA), aids the reader in understanding.

Throughout this chapter the term *installation* is used to refer to *software* installation. Hardware installations are not covered in this chapter.

11.2 Goals

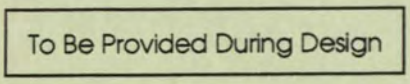
The goals of OZIX system directory layout and installation are:

- Minimize service interruptions when software is installed or updated.
- Maintain system integrity during the installation period.
- Eliminate the need for coordinated releases of OZIX and layered applications because of version dependences.
- Simplify the installation procedure; the procedure should not preclude customer installability.
- Layered application installation procedures should be non-proprietary, so that third parties and customers may use these same procedures to install their applications.
- Installation of the base system should take less than one hour.
- Meet the priority 1 requirements as outlined in the *Preliminary OZIX Version 1.0 Serviceability Requirements Document* (SRD) [2].
- Provide compatibility with existing Digital ULTRIX third-party installation procedures.
- Provide remote, layered-application installations to and from OZIX.

11.3 System Directory Layout

OZIX system directory will follow the ULTRIX model as illustrated in Figure 29.

Figure 29: OZIX System Directory Tree



Additionally, the directory layout will support compartmentalization of log files as required by the SRD.

11.4 System Installation

The primary goal of initial system software installation is the overall time to install. To accomplish this the following basic steps are used to perform an initial system installation:

1. A minimal OZIX system is booted from the distribution media.
2. All necessary questions are presented to the user. Questions include node name, address, initial passwords, and system configuration data. On line help will be provided for all questions and defaults will be applied where appropriate.
3. Those parts of the system directory structure requiring read/write access by the running system are created and populated. This represents less than 15% of the total data manipulation required for system installation.
4. The remaining read-only data in the system directory structure resides on the distribution media. This represents the remaining 85% of data manipulated by the system installation procedure.
5. The minimal OZIX system used in steps 1 through 3 is terminated and the full OZIX system is started. The full system uses the read/write area created in step 3 above and the read-only data on the distribution media.
6. At this point a full OZIX system is available for full functional use. Although performance may be temporarily limited by the access speed of the distribution media, the system operates in a normal fashion, obtaining read-only data from the distribution media and applying updates to the read/write area created in step 3 above.
7. Since the distribution media may not have an acceptable I/O rate, the data in the read-only areas on the distribution media is transferred to a "real" system directory. During this time, the system is still fully functional.
8. After the data has been transferred, the distribution media is removed and system operation continues from the real system directory.

In general, system installation requires only one reboot of the system. Installation time is reduced to the amount of time it takes to build the read/write data on the system. The time it takes to transfer the read-only data, which accounts for about 85% of the data, is not included in the installation time since the system is fully usable while this transfer is taking place.

11.5 Application Installations

In order not to create another installation procedure with which applications must conform, the ULTRIX setld utility will be ported to OZIX. Refer to the ULTRIX documentation set for details. Applications conforming to setld software distribution technology will not have to be modified to install on OZIX.

In addition, remote software installation support will be provided by porting the ULTRIX Remote Installation Service (ris). Refer to the ULTRIX documentation set for details.

11.6 Internationalization

System installation and update are fully internationalized. If possible, some of the internationalization profile will be derived from settings in the hardware. If not possible, the installation procedure will have to query, in English, for the profile.

Internationalization of layered-application installation is limited to that which is provided by setld and each individual application's installation procedure.

11.7 Related Documents

[1] M. A. Ditto, *OZIX System Administration Overview*, Version 1.0, September 27, 1989.

[2] T. Baker, J. Bingham, M. Licata, D. Miller, T. Siebold, L. Stivers, *Preliminary OZIX Version 1.0 Serviceability Requirements Document*, Revision 0.9, August 4, 1989.

CHAPTER 12

PERFORMANCE ENGINEERING

12.1 Abstract

This chapter presents a high-level overview of the instrumentation and performance analysis methodologies for OZIX, and the services provided to accurately characterize its performance and the performance of applications it supports.

12.2 Introduction

In general, the extensive use of performance instrumentation and modern analysis techniques in parallel with software design is often overlooked by developers. More commonly, designs are completed and if unacceptable performance is observed during their testing, steps are taken to identify and correct the problem.

To meet the needs of the production systems market, however, a comprehensive performance strategy must be implemented from the beginning of the software design, rather than after the fact. Production systems markets require comprehensive instrumentation and performance analysis in order to provide the information necessary to enable effective capacity planning, configuration management, system management, performance characterization, and to support transaction processing in general.

This chapter presents our strategy for integrating performance analysis into the earliest phases of software design, generally, and into OZIX specifically. This chapter provides some background on OZIX performance methodologies, and then describes OZIX instrumentation and data collection. This chapter also describes the performance characterization tools for OZIX, and the OZIX modeling strategy.

12.3 OZIX Performance Engineering Goals

The Performance Engineering goal is to ensure that OZIX exhibits leadership performance when compared to systems offering comparable functionality and robustness. Our role is to define the metrics, develop the methodologies and tools, and support the designers in the achievement of this goal. Specifically, this translates into the following set of goals:

1. To provide OZIX developers, Independent Software Vendors (ISVs), and end users with the set of leadership tools and instrumentation services necessary to analyze and improve system and applications performance.
2. Establish and characterize clear and valid performance measurements that illuminate the differences in performance between OZIX-based systems and other systems whose operating system is of comparable scope.
3. To develop a leadership capacity planning strategy.

12.4 OZIX Performance Methodologies

This section provides an overview of how OZIX performance will be measured and characterized.

12.4.1 Workload Characterization

In order to assess OZIX component performance we need to have on hand a set of *standard* workloads with which we will study the contribution ratios of OZIX and OZIX components, and the equilibrium kinetics of OZIX-based systems.

We propose, initially, to acquire the following workloads (and benchmarks):

- The USAA, DuPont, Batelle, and Citibank workloads

  These workloads were chosen because of the detail in which they are described and their commercial nature. These workloads are sufficiently different, especially in their I/O characteristics‡ as to constitute a broad suite of complementary workloads.

- The Debit/Credit, Warehouse Inventory Control (WIC), and GE NFS benchmarks

  These Benchmarks are of little design interest to engineering. They command a high level of interest to the corporation because, for better or worse, customers make buying decisions based on the results of these benchmarks.

Using these workloads and benchmarks, we will apply the methodology developed for the scalable weighted application test (SWAT) kernels to generate synthetic and parameterizable performance kernels. The SWAT kernels are composed of two sets of kernels: The first is a suite of some 25+ synthetic routines, each of which is designed to exercise a specific attribute of a computer's CPU and memory complex. The second is a set of computational kernels derived from 5 different user applications. By running each suite, data is produced that may then be analyzed to determine those computer attributes that correlate most closely with application kernel performance. Technically, the methodology used to establish the sensitivity and correlation matrices, is general linear least squares analysis using singular value decomposition. This approach will enable us to more easily emulate a customer's computing environment; The load imposed by the customer's usage patterns on their systems.

As OZIX evolves, this approach should allow us to spend more time characterizing the performance of OZIX and less time acquiring and running workloads.

12.5 OZIX Instrumentation and Data Collection

12.5.1 Overview

The extent to which designers and users are able to critically evaluate the system performance of OZIX-based systems, and the extent to which such evaluations lead to meaningful and accurate measurements, constitutes a powerful incentive to prefer OZIX-based systems over the competition. To achieve this, OZIX needs a consistent, low-overhead, and comprehensive set of instrumentation and data collection services that applications and system software can employ to this end. To ensure that OZIX meets these requirements we will establish a consistent policy for instrumenting OZIX and its components, i.e., an *Instrumentation Architecture*. We will design a data collection facility, a low-level software component, that will enable the system to collect component and system-wide data; the data collection facility will conform to the entity management architecture (EMA) to provide consistency of access.

‡ Specifically, transfer size, request rate, physical locality, and file congestion

The instrumentation architecture must support two data collection methodologies—Event detection and sampling.

12.5.2 Event Detection

An *event* is a change in an object's state. *Event detection* is the process of dispatching and recording information regarding events.[1]. Thus, event detection is a technique to be used when it is necessary to know a sequence of events or the exact number of their occurrence over a given interval of time. Most capacity planning applications, for example, use event collection to produce performance and utilization traces to drive a model of the system in order to predict future requirements.

OZIX will employ *software probes* for event detection. A software probe is a recording routine dispatched upon the occurrence of an event. A *hook* is inserted into the code at a point such that when the event occurs control is transferred to the probe. After recording the event, and possibly other information, control is returned.

OZIX must have event detection capability carefully integrated into its component designs. If event detection is not used judiciously, or if the software probes are not placed with great care, system performance can be greatly degraded. Moreover, OZIX system management will be employed to control selection of event collection. Together these attributes ensure that event detection is implemented with the least cost to performance and access to event detection services is consistent.

12.5.3 Sampling

Sampling is a statistical technique usable whenever the measurement of all the data characterizing a set of people, objects, rates, or events is impossible, impractical, or too expensive. At regular intervals system- and/or component-specific counters are read. Sampling is commonly used to observe rates and frequency of events. It is also used to obtain a snapshot of the current state of the system at some particular point in time (i.e. number of users, amount of free memory remaining, etc). The process of data collection by sampling causes less disturbance to the system then event detection by software. It is less accurate then event detection, but can be made sufficiently accurate if the samples collected are reasonably large.

Sampling is used to measure the fractions of a given time interval each system component spent in its various states. The data collected during the measurement is subject to analysis to determine what happened during the interval, in what ratios the various events occurred, and how the different types of activity were related to each other.

12.5.4 OZIX Instrumentation Data Collection Facility

The OZIX Instrumentation Data Collection Facility *IDCF* is responsible for collecting performance and utilization data maintained by OZIX. The IDCF will provide data drawn from various levels of the system. Data collected by the IDCF will be categorized as follows:

- Subsystem-specific data
- OZIX-specific data
- Digital-specific data[1]
- Distributed-system specific data

[1] The point at which an event is detected is also known as a trace point. The process of event detection is commonly referred to as tracing

[1] Digital specific information pertains to specific Digital hardware or software. For example, Disk I/O statistics

The IDCF conforms to the common collection facility architecture being developed for all three of Digital's operating systems, ULTRIX, VMS, and OZIX. This will enable third-party vendors, ISVs and other groups in Digital to provide tools for understanding distributed system performance, dynamic load balancing, system configuration and distributed capacity planning. For more information see the OZIX Instrumentation Collector Functional Specification .

12.5.5 OZIX Instrumentation Services

As part of the instrumentation architecture OZIX will provide services for control of, and access to, OZIX instrumentation. The instrumentation services will conform to and employ OZIX system management. The instrumentation services will satisfy the DECxTP instrumentation requirements and will provide the foundation for the OZIX capacity planning strategy.

An example of some of the instrumentation services provided by OZIX:

- Access and control of counters for sampling (real-time) information
- The registration of probes for event detection
- Activation and deactivation of probes

12.5.6 OZIX Instrumentation Architecture

The following figure illustrates a conceptual, high-level overview of the OZIX instrumentation architecture.

The instrumentation architecture specifies the flow of data from the lowest levels of the OZIX system via manageable entities to the user. From a functional perspective the components are:

- The Management entities are instrumented components whose instrumentation is specified by their designer according to the OZIX Instrumentation Guide for Developers.
- The EMA backbone (including the Event Dispatcher) provides access to the instrumentation provided by the manageable entities.
- The OZIX Integrated Data Collection Facility (aka Data Collector) is responsible for collecting the data from the manageable entities.
- The Common Collector is an abstraction that provides a uniform mechanism by which to access operating system instrumentation. The services will be uniform across multiple operating systems (e.g., VMS, OZIX, ULTRIX, and OSF/1).
- Various data analysis and resource utilization applications can make use of the information collected by the Common Collector.
- The Presentation Management layer provides a set of presentation services (e.g., graphical visualization) to the user.

The Performance Group is working with representatives from VMS, ULTRIX, and DECcp, among others, to provide a mechanism by which performance and other data can be presented uniformly across a distributed, heterogeneous environment. Currently, the Common Collector will be supported by OZIX, ULTRIX, VMS, and OF/?.

Figure 30: The OZIX Instrumentation Architecture

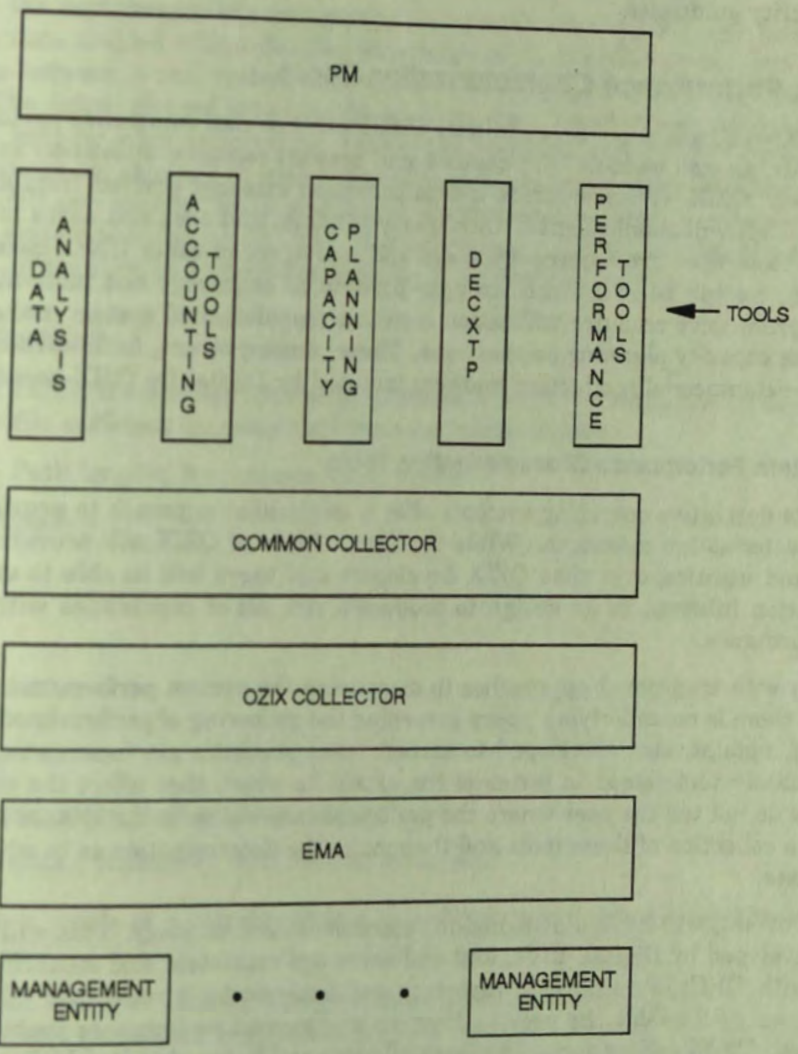
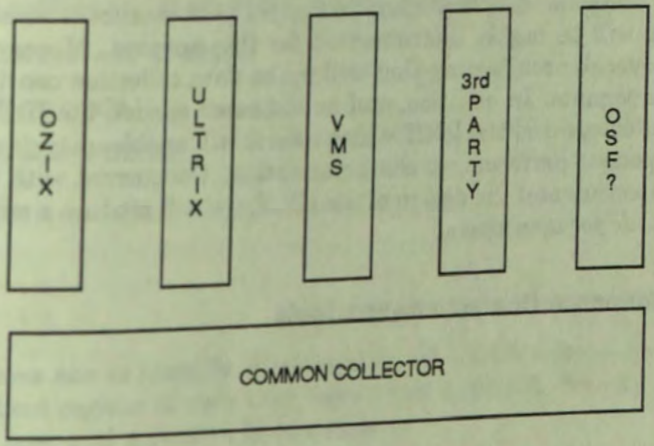


Figure 31: The Common Collector Facility



12.5.6.1 Security Implications

The OZIX instrumentation collector and the common collector will be designed in accordance to the OZIX B2 security guidelines.

12.6 OZIX Performance Characterization Tools

Initially OZIX will have a set of traditional UNIX utilities and commands including profilers and timing utilities, as well as tools that capture and present resource utilization information. Unlike UNIX, however, OZIX will incorporate a data collection strategy derived from an instrumentation architecture. This will enable Digital, third-party vendors, and even end users to build much better performance tools than are enjoyed by users and managers of other UNIX-based systems. Using this approach, we will be in a much stronger position to encourage and motivate the development of more comprehensive resource utilization monitors, sophisticated system management tools, and more accurate capacity planning applications. These, among others, fulfill critical requirements for success in the commercial production markets targeted by Digital for OZIX-based systems.

12.6.1 System Performance Characterization Tools

UNIX and its derivative operating systems offer a minimalist approach to providing tools to obtain system characterization measures. While the first release of OZIX will provide an identical set of commands and utilities, over time OZIX developers and users will be able to exploit the extensive instrumentation inherent in its design to produce a rich set of capabilities with which to measure system performance.

The problem with traditional approaches to measuring the system performance of UNIX-based systems is that there is no underlying policy governing the gathering of performance statistics. The tools (e.g., vmstat, netstat, etc.,) developed to extract these statistics are inconsistent, often incomplete, and inadequately understood in terms of the extent to which they affect the measurements themselves. They do not tell the user where the performance problem in the system is. It is left up to the user to run a collection of these tools and then make the determination as to where the performance problem exists.

OZIX, by contrast, will have a designed-in instrumentation strategy. This will help to ensure that the tools developed by Digital, ISVs, and end-users are consistent and meaningful. In addition, we will work with DECxTP to develop reporting and analysis tools which use advanced visualization techniques that will enable the user to diagnose and correct performance problems. To ensure compatibility with UNIX and its derivatives we will also provide the standard UNIX system activity tools and other standard functionality (e.g., vmstat, netstat, fstat, sa* etc.,).

In order to characterize a components contribution to system performance it must be instrumented. OZIX, and its components will be highly instrumented for this purpose. Moreover, the instrumentation will conform to an overall architecture that will make data collection consistent across OZIX-based systems and its components. In addition, and as indicated earlier, the IDCF will be provided. The instrumentation architecture and the IDCF which uses it will enable us to develop new and more appropriate tools for component performance characterization. Concurrent with the development of the instrumentation architecture and the design of the IDCF, we will produce a supporting document, *OZIX Instrumentation Guide for developers*.

12.6.2 Component Performance Characterization Tools

12.6.2.1 Elapsed Time Analysis

To fully characterize the performance of a component under evaluation (CUE), only an analysis of its elapsed execution time coupled with a detailed knowledge of its *execution profile* (see below) can yield the relationship between it and system performance. For example, a CUE may execute in N CPU microseconds. The actual elapsed time may be 10 times that, however, because it spends time waiting for resources held by other components. One of the problems with "UNIX" PC sampling is that the data it provides is not obtained in such a way as to indicate to the developer *what the system was doing while the CUE was waiting*. OZIX will provide services that will help a software developer determine where the system is spending its time, or what resources are being held, while a CUE is blocked. This is vital information to a developer wanting to enhance performance.

12.6.2.2 Execution Profiling

An execution profile of a CUE is a detailed time-ordered record of events detected during its execution. A typical execution profile sufficient for design optimization might contain:

- Instruction data - Path lengths, frequencies, basic blocks
- Elapsed time information
- CPU cycle usage
- Call graphs
- Register usage
- Memory references
- Istream references
- Lock statistics - Locks attempted, acquired/released, etc.
- Device I/O request data - Frequency, size, read or write, etc.

Generating the execution profile of a component is a necessary step in determining its impact on overall system performance.

Over the lifetime of OZIX it will be necessary to provide such services and support (e.g., documentation, training) as to enable developers to generate detailed execution profiles. Accordingly we will provide these services and a set of tools necessary for OZIX designers and other OZIX software developers to critically assess their component's impact on system performance. The services necessary to generate execution profiles and those which we will provide are:

- General event detection and collection
- *In situ* basic block coverage
- Non-invasive instruction tracing
- *In situ* I/O tracing
- Cycle counting

12.6.2.3 Coverage

Coverage analysis allows one to identify what sections of a CUE were executed. This allows the developer to identify dead regions of code that were never executed. We will provide tools for basic block coverage of both user and executive level software.

12.6.3 Initial Performance Characterization Tools

As previously stated OZIX will initially support the traditional UNIX tools. These tools are used by system administrators to determine how the OZIX system is performing, by system administrators and software developers to determine a components contribution to system performance and by software developers to characterize their components performance and to provide coverage information[1]. OZIX will initially provide the following tools:

- System characterization tools (these tools are also used in determining a components contribution to system performance)
  - SA* - System activity reporting tools
  - ipcs,pstat,vmstat,iostat,netstat - interprocess communication,system, virtual memory, I/O and network statistics
  - Instrumentation services - To provide control and access to OZIX instrumentation. (These are defined in the (*OZIX Performance Architecture Guide*)
- Component characterization tools
  - Pixie, pixstats - Pixie instruments a component, pixstats reports the results
  - Prof - PC sampling and coverage reporting tool
  - Time - timing information
  - ps - process status

These, among others, constitute the basic services necessary to obtain overall system utilization and computational capacity information - so vital for system management and capacity planning functionality. The quality of these tools will depend heavily on the instrumentation services provided by OZIX which is precisely why they are being incorporated as a part of its basic design. As it is the integration of instrumentation services and system tools that are traditionally referred to as software monitors, a key part of the OZIX strategy will be to work with third-party vendors to develop such monitors; perhaps with advanced visualization techniques that will enable even non-experts to characterize and manage system performance.

All the performance tools produced by the OZIX performance group will conform to the OZIX internationalization guidelines. Any new tools that require a command line or shell interface will conform to the OZIX command & shell style guidelines.

12.6.4 Capacity Planning

Capacity planning is the process of predicting and planning computing resources to meet business growth requirements or anticipated needs. OZIX will have an industry leading capacity planning strategy. That strategy is based explicitly on providing a leadership operating system for capacity planning applications. Leadership means that the OZIX services provided to capacity planning applications meet *all* of their instrumentation requirements. Moreover, for OZIX to be considered leadership in this area the capacity planning applications that run on OZIX must produce more accurate and more useful results than applications running on other operating systems. The quality of these particular applications and the value OZIX customers place on them are the final measure of leadership in this area.

[1] A component is some executable code (i.e. a program, an OZIX subsystem or shared library).

The fundamental requirements in providing a capacity planning application, and how OZIX will meet them are:

- Instrumentation

  Relevant data collection is the first, and in some ways, the most important requirement in capacity planning. The OZIX IDCF will provide capacity planning applications a comprehensive yet low-overhead mechanism to conduct event detection and other relevant statistics.

- Data analysis

  Data analysis routines within a capacity planning application make little use of operating system services. They do require, though, that the data collected via the operating system's instrumentation services are able to be abstracted into workloads that are meaningful (and valid) to the users. For example, "users per day", "number of terminals supported", and so forth.

- Modeling and Prediction

  A model of the system is then generated (usually employing a combination of simulation techniques for accuracy and analytic techniques for speed) and validated against the utilization data created during the data analysis phase. The anticipated workload is then parameterized and run on the model and the results are used to extrapolate to a new configuration.

  This is a critical phase of the capacity planner. The accuracy of the model is the principal measure of its quality. The best capacity planning tools on the market today are accurate plus or minus 30%. It is an OZIX goal to be accurate to within 20%.

  Configuration prediction will be based on transaction response time and/or system throughput as a function of the user's anticipated system load. For example, if the enterprise anticipates the number of users to grow by 20% per year over the next five years what additional capacity will the enterprise be required to purchase in order to maintain existing levels of performance?

- Reporting

  The reporting software must present meaningful results to different levels of personnel; This might include MIS managers, their technical staffs, and division, even CEO and CFO level executives. It is critical that OZIX not make it difficult or otherwise impede the generation of high quality, and easy-to-comprehend reports.

The implementation of the capacity planning architecture will involve work by OZIX developers, Digital' Capacity Planning Engineering Group, and most especially third party capacity planning software vendors. The requirements for the OZIX capacity planning strategy will be provided and driven by OZIX Performance Engineering with support from OZIX product marketing and other organizations as necessary.

12.7 OZIX Modeling Strategy - Overview

Our modeling strategy is to provide two distinct, but integrated levels of modeling support. The distinction between the two levels is made according to whether a *simulation* model or a *simulator* model is employed in any particular OZIX performance study. This is determined by the extent to which the execution of the system, or system component, must be explicitly represented. The two levels discussed here, which form the cornerstone of the modeling strategy are, the creation of both simulation and simulator models.

1. *Simulator Models*

Simulators are particularly useful for component performance characterization. A simulator is simply an abstraction of the hardware architecture implemented in software. Its principal advantages are, but not limited to:

- It can execute code in advance of the existence of real hardware.[7] This allows critical component performance studies to be conducted early in advance of the design and test phase of OZIX.
- Performance collection within a simulator is non-invasive. This means that a component under evaluation does not necessarily have to be instrumented in order to collect measurement data. Therefor the CUE runs as it would in a real environment.
- A simulator may be designed to produce performance statistics not practical in a real environment or without various types of hardware monitors (i.e., cache and/or bus monitors, instruction counts, cycle counts etc).

We will provide simulator support for OZIX developers by extending and/or redesigning existing simulators (e.g., Sable). This will be particularly important to developers working on early baselevel components.

2. Process-oriented Simulation

Simulations are useful to OZIX developers for characterizing system performance (throughput, response time, etc.,) as a function of operating system design. A process-oriented performance simulation is a software program in which processes are created which make requests of other processes (e.g., OZIX components) and/or hardware platform components (models of the CPU, memory subsystem, etc.) for resources, scheduling, and other services.

We are prepared to provide and support process-oriented simulation programs, tools, or support that focuses explicitly on network and I/O performance problems.

12.7.1 OZIX Modeling Goals

Our overall goal is to provide modeling tools, services, and support to OZIX component and system designers. This overall goal is our way of serving a shared overarching goal, i.e., to insure that OZIX achieves the very highest performance that technology, product constraints, and engineering skill is able to deliver.

The goals of our modeling effort are:

- To enable the characterization of the performance of individual OZIX subsystems, and the OZIX executive without the need for actual hardware.
- To enable the characterization of the performance of OZIX as a function of the underlying hardware architecture.

12.7.2 OZIX Modeling Strategy

For high level OZIX performance studies, we will design a simulation model of the OZIX executive. This model may be designed from existing models when one exists to do the job (i.e. a network model). Initially we will use this tool study such questions as:

- System performance prediction as a function of the layout of executive subsystems.
- System performance prediction as a function of I/O software design and the underlying I/O component configuration.

[7] A compiler capable of generating target hardware instructions is, of course, required.

- System performance prediction as a function of network interconnection and flow-control strategies.

The simulation model will be written in C, will employ the CSIM run time library.

For performance characterization studies requiring the explicit execution of subsystem code, we propose to build or extend existing architectural simulator. The simulator will provide an execution environment for detailed characterization of subsystem performance (as well as program development/test, fault generation/error recovery, and portability). The simulator will provide the following features:

- Target instruction set and cache layout
- Multiprocessing
- 32 and/or 64-bit virtual addressing
- Extensive instrumentation
  - Accurate CPU timing (via cycle counting)
  - Exact instruction counts
  - Basic block counting
- Integration with Ladebug, Prof, and other commonly used UNIX tools.
- Instruction set independence

Visualization tools will be constructed where appropriate.

CHAPTER 13

SOFTWARE QUALITY AND TESTING STRATEGY

13.1 Overview

This chapter describes:

- Each component of the quality and testing strategies and the role the component plays in the overall strategy
- The motivation, requirements, goals and nongoals behind the extensive and aggressive software testing strategy to ensure we are building a quality product
- The components and capabilities that must be delivered by various engineering groups to support the quality and testing strategies

13.1.1 Motivation

The product goals for OZIX demand that the system provide 7x24 availability, conform to industry standards (POSIX, OSF, X/OPEN, etc.), and be extremely reliable, secure, and robust in order to be competitive in the production systems marketplace. Quality, reliability, and security must be designed into the system in addition to thorough testing of the completed system for the necessary levels of robustness.

In defining the testing strategy for the system, we have attempted, where possible, to utilize component strategies that provide the greatest *coverage* for the lowest cost. This chapter defines a composite testing strategy, specifying a variety of testing techniques and tools to be employed to obtain the highly reliable, robust, error-free products required.

A thorough, effective test system allows us to make corrections or enhancements to the system and subsequently run the test system against the updated software:

- To ensure that no existing functions were broken as a result of the change
- To permit development of new tests to cover the changes for future development
- As a regression test to ensure that the same problem isn't reintroduced into the system in the future

13.1.2 Software Quality and Testing Strategy Requirements

The quality and testing strategy depends on:

- Software designs and implementations which are high-quality, well-engineered and well-integrated

  Without well-integrated, well-engineered designs, we waste precious resources tracking down logic, design and integration errors that should have been caught during the design and implementation processes.

- Testing methodologies and tools to:
  - — verify that the designs are implemented as specified
  - — verify where possible that all erroneous inputs and data, singly or in combination, are detected and handled properly
  - — exercise and load the system to determine the interactions of components and the effects on the system of depleting resources

13.1.3 Software Quality and Testing Strategy Goals

The goals of the quality and testing strategy are to:

- Identify methodologies and procedures to introduce quality, reliability, security, usability, and maintainability into the system before implementation and testing is begun
- Define methodologies, tools and guidance for developers building and executing test sets
- Build test systems which:
  - — Confirm that the system behaves correctly under specified and adverse circumstances
  - — Provide a structure into which unit test procedures, error verification procedures, and so on can be easily added
  - — Eliminate the reoccurrence of "fixed" problems through regression tests
- Provide a measurement of the code coverage attained by the test system
- Provide a method to track bugs and fixes, and to enhance the test suite to prevent the same bug from being reintroduced into the system

System testing is an ongoing activity, not a point event. The testing strategy must provide plans for the life cycle of OZIX.

13.2 Software Quality and Testing Strategy Components

The components of the software quality and testing strategy are described in the following sections.

13.2.1 System Specifications

The design of OZIX is being done in a formalized manner, with the end result being a functional specification, an interface specification and a detailed design specification for each component of the system.

The formats for all of these documents and the document review process is outlined in the *OZIX Software Development Procedures*. The following paragraphs summarize the documents and their review procedures.

The functional specification is a high level summary of the component functionality. The functions it describes will map to the interface specification for details, to the detailed design specification for implementation, and to the code for function performance.

After the functional specification for a component has been completed, it must pass two reviews. The area review (performed by a designated team of developers known as area reviewers) determines if the specification fits within the overall vision for the project, delivers the required capabilities, and is complete.

The primary review is performed by team members, area members, and members from other areas. The specification is reviewed for its technical content, completeness, and quality of the interfaces presented.

After the functional specification passes primary review, it is put under change control and is made available to the entire software group.

The interface specification for each system component describes all external interfaces to the component. All externally visible data structures are also documented. Using this information, developers of other components are able to successfully write code that invokes this component.

The interface specification must pass one review which may be combined with the detailed design review at the discretion of the project leader.

The detailed design specification for a component contains the logical module designs. This chapter is reviewed by the project team for design validity and completeness, and put under change control. It is primarily used by the project team during the implementation phase to produce the code for the documented component. It is also valuable during project maintenance, for training of new project members, for creating user documentation, and for communicating the design to external groups.

13.2.1.1 Benefits

Producing the functional, interface and detailed design specifications is an enormous amount of work. However, the benefits of this effort far exceed the effort invested:

- The comprehensive design specification and review process help ensure the quality, consistency, coherency, completeness, and thoroughness of the design
- The detailed designs are used to build *white box* tests for integration testing and base level testing of system components
- Over time, as new team members are brought into the software group, they are able to read a formal document describing the system, rather than being forced to learn the design by tediously reading the code
- The documents can be used by layered product developers to gain a deeper understanding of the system, reducing the need for interaction with members of the OZIX development team
- The documents can be used to maintain the code and to aid in design of future releases of OZIX
- The documents can be used for the development of training materials for product support

13.2.2 Code Reviews

As the development work turns to coding, we must ensure that the code is of the highest quality, and is consistent across the product. One way in which this will be accomplished is via code reviews. The following paragraphs summarize code reviews. More details and pointers to other documents on code reviews can be found in the *OZIX Software Development Procedures*.

A code review is a good mechanism for information exchange among developers. It is also a good method to catch subtle bugs that a developer may be too close to the code to see. The earlier a bug is caught, the less expensive it is to fix.

The goals of a code review are to check:

- Adherence to coding standards
- Correct solution of the problem
- Correspondence of code to design

- Correct use of language features
- Maintainability of the code
- Understandability of the code
- Efficiency of the code

The coding standards for the code developed on this project are defined in the *OZIX Naming Standards and C Coding Conventions* document.

Although it is ideal to have every line of new code reviewed, it is usually impossible to do so. All code which makes up the *trusted computing base (TCB)* for *B2* security must be reviewed. Interface routines, asynchronous routines, frequently called routines, exception handlers, time critical routines, hardware error routines, and complex routines should be reviewed. It is up to the project leader to determine which code will be reviewed if all the code cannot be reviewed. Complex routines should be walked through line by line to detect logic or coding errors.

Enhancements made to ported code should also be reviewed. The project leader and developer decide on the amount of code needed for the review to provide adequate context for the enhancements. Enhancements to ported code should follow the style of the original code unless a new module is created. In this case the code in the new module should follow the *OZIX Naming Standards and C Coding Conventions* as much as possible.

13.2.2.1 Benefits

Effective code reviews done early in the development process promote a higher quality product by:

- Pinpointing problematic code
- Ensuring that all code is developed using the same coding style. This promotes more readable software, and over the long term, aids in reducing maintenance costs.
- Providing a better understanding of effective coding constructs
- Providing a review of the validity of the implementation and correspondence with the design
- Familiarizing developers other than the author of the module with the code
- Providing feedback to other developers, and potentially, the compiler group
- Detecting algorithmic errors in complex modules at earlier stages of development, reducing time lost later in the project

13.2.3 Unit and Integration Testing

Developers are responsible for the unit testing of components and testing the integration of components. The purpose of this testing is to give the developer and project leader reasonable assurance that the implementation is error free.

The Software Test Team develops templates for unit tests and routines to generate reusable test cases to assist the developers in writing comprehensive unit tests for their code.

The test cases are developed using *equivalence partitioning* and *boundary value analysis* to ensure the greatest test coverage at the least overall cost. These test selection methods are described in detail in G. J. Meyers' *The Art of Software Testing*. Test cases are structured to detect sensitivities in parameter presence, range of legality, and combinations of problems.

This framework provides a mechanism for individual developers to easily add unit test code to the test system if appropriate. The tests developed during unit and integration testing may be added to the test system to be executed on all new system base levels to ensure that properly functioning code continues to execute correctly and that expected errors are detected.

Over time, the tests developed during unit and integration testing may be added to the regression test suite to increase test coverage. The Software Test Team builds the basic test case library, regression test suite, and develops a "how-to-use cookbook" to allow developers to add tests more easily to the test suite.

13.2.3.1 Benefits

The regression/unit test framework:

- Relieves the developers of the tedium of building the test framework, allowing them to concentrate on the tests, hopefully allowing more tests to be built
- Provides a flexible, extensible structured framework on which a complete unit test system can be built

13.2.4 Executive Testing

Testing is performed on the OZIX Executive code (*nub* and executive *subsystems*) by both the developers and the Software Test Team. Tools used for building unit tests are utilitized for building base level tests. Many unit and integration tests for the executive become part of the base level regression test suite.

Building the base level test suite is an iterative process that involves constructing the tests, evaluating the results, correcting any problems found, and enhancing the test suite for additional coverage. Tests are also added at each base level to test all new functionality added to the executive for that base level.

Time is allocated in each system build to permit execution and verification of the base level test suite.

13.2.4.1 Benefits

The testing of OZIX executive code:

- Exercises the executive code more thoroughly than can be done at a higher level
- Provides better coverage than tests executed at user mode
- Finds problems earlier than normal in the development cycle of the system

13.2.4.2 Nub Routines Test Suites

Test suites are being developed for the nub routines. System level hooks are used in combination with debuggers and hardware simulators to run these tests before application level functionality is available on the system.

Tests for the executive interface routines are developed using both white box and *black box* approaches. The code is examined and the tests are written to cover as many code paths as possible. Boundary conditions and error handling are stressed in particular.

Since routines change and new ones are added at each baselevel, the tests are also appropriately modified by the developers and the Software Test Team.

13.2.4.3 Executive Subsystem Tests

Subsystem tests are written by the developers with assistance from the Software Test Team. These are developed using both white box and black box approaches. A method to simulate errors which are not easily generated for testing is being investigated to assist in the thorough testing of error paths in the subsystem.

All security testing, device-related error testing and any required fault insertion is performed by the developer. Where possible, tests are added to the regression test suite. The Software Test Team is working with the appropriate groups to develop a detailed strategy in this area.

13.2.5 Application Level Testing

Application level testing involves testing of the *API* and application subsystems including commands, utilities, system services, and library routines. Porting the *ULTRIX Test Executive (UTE)* to OZIX for this testing is being investigated. Test suites developed by UEG for ULTRIX are modified for testing OZIX specific features as necessary. We are investigating the feasibility of modifying the POSIX conformance tests to run under UTE.

13.2.5.1 Benefits

The user mode test suites:

- Promote building a maintainable, extensible highly-structured suite of tests for runtime routines
- Automatically test ported code, because the tests for this code will also be ported
- Share testing code with the ULTRIX/OSF group

13.2.5.2 Commands and Utilities Test Suite

A test suite is being developed to test the commands and utilities available on OZIX. The test suite is based on the UTE commands test suite with modifications and enhancements made for OZIX specific features.

13.2.5.3 API Subsystem and Library Test Suite

Outside of the normal unit and integration testing done for these components, a test suite is being developed to test the external interfaces to OZIX. This test suite is based on the UTE system services and functions test suites with modifications and enhancements made for OZIX-specific features.

13.2.5.4 DECwindows/Xwindows Test Suite

Currently an Xlib validation suites exists for ULTRIX, but no automated procedures exist for capturing and comparing screen images. An OSF version of *DEC/Test Manager (DTM)* is planned for the same time frame as OZIX FRS, but it will not be available during most of the OZIX development. A method to record and playback user input on all types of workstations is in the process of being developed by OSG. This tool (currently named Xigor) depends on the Xtrap server extension being installed with the Xserver. The script files generated by Xigor are planned to be in the same format as needed by the OSF version of DTM making it a viable tool to use during OZIX development although we will have to add screen image capture and comparison to it.

13.2.5.5 AIA Routines Test Suite

A test suite for all OZIX-supported *AIA* routines is being developed to run under UTE.

13.2.6 Base Level Regression Testing

Tests are run on each base level of the system immediately after it is built. These tests are a combination of unit and integration tests from various components, user mode tests, and system integration tests. The base level build is not considered ready for general use until all tests complete successfully.

The regression tests are run only after an initial stable base level has been developed and tested using the unit and integration tests for the OZIX executive.

13.2.6.1 Benefits

The benefits of base level testing are:

- Verify correct system build
- Early identification of system integration problems
- Release of stable base level software

13.2.7 Problem Tracking

Starting with base level regression testing and continuing throughout product development and delivery, all bugs are tracked using a problem reporting system which is expected to be integrated with the project CASE environment, if at all possible. If the bug was not identified through testing, then, if possible, a test is added to the test suite to ensure that the bug doesn't reoccur. The correspondence between the bug report, the bug fix in the code, and the test are maintained. Changes to documentation and code (whether they result from bugs or not) must be done via that change proposal mechanism described in the *OZIX Software Development Procedures*.

13.2.7.1 Benefits

Problem tracking:

- Identifies what problems have been fixed and which still exist
- Builds a better regression test suite
- Ensures that fixed bugs do not reoccur

13.2.8 Test System Coverage Analysis

In order to verify the effectiveness of the test system coverage, we must be able to determine the extent of the system code exercised by the test system. The profiling tools developed by the performance group will be used when possible.

The Software Test Team analyzes the coverage information obtained during the execution of the full test system. They then develop a prioritized list of untested code paths which are candidates for fault insertion testing, or determine what automatic test can be developed to cover the paths.

The goal is to use the test system to exercise 80% of the code.

13.2.9 Full-time and Stand-alone Usage of the System

As soon as the developing system is capable, software system builds, other test suites, and application suites are run frequently on in-house development systems in order to maximize system usage and to determine system behavior when running with little idle time. We anticipate that this load should exercise the system and provide beneficial feedback without affecting users very much.

A moderate amount of testing is necessary on dedicated machines for areas such as power failure, hardware errors, system crashes, low-level system debugging, system and configuration management, etc. This testing is scheduled and users have access to stand-alone machines for this purpose in order not to impact users on development systems or be impacted by system activity.

13.2.10 OZIX System Installation Verification Procedure

The OZIX system installation verification procedure (IVP) is a required component of the software kit. It is executed as part of the system installation procedure to ensure that the system has been properly installed and consists of the System Exerciser running under the On-line Diagnostic Monitor (see Section 13.2.21) and a selected subset of software tests.

13.2.10.1 Benefits

The software IVP:

- Reduces support costs by ensuring that a correct installation was accomplished
- Increases the probability that the installation will be trouble-free
- Provides a product acceptance test

13.2.11 Field Test

Field test is the engineering development process by which the total hardware/software/support system is exercised. It provides a generally friendly environment in which the system is utilized heavily in a nonproduction mode to shake out problems not detected or impossible to shake out in the in-house environments. Field test sites are selected based on criteria specified by engineering, to fulfill specific needs such as applications used, hardware environment, and so on.

Field test is an engineering activity. It is not intended for marketing or sales use. It is not a mechanism by which early systems can be placed into customer installations, unless this placement serves the needs of engineering.

Preliminary field test requirements and plans will be issued as part of the Phase 1 documentation.

The feedback loop, typically through a Quality Assurance Report (QAR) system, will be implemented such that there is no additional time or levels of bureaucracy introduced between the submission of the problem report and its receipt by engineering, permitting engineering to respond promptly to the reports.

13.2.12 Software Documentation Testing

We must ensure that the software documentation accurately and completely describes the software/hardware product that is shipped. This is done by extensive review of the documentation by the engineering group, by internal users, and by our field test sites. It is planned that several of the field test sites will be contacted during field test to determine the accuracy and usefulness of the documentation.

The individual writers are responsible for ensuring that their manuals have an effective review and are reviewed by the responsible engineers. Each manual has a key developer responsible for the technical accuracy of its contents and technical signoff.

13.2.13 Stress/Load Testing and Subsystem Interaction

Stress and load testing aids in defining the operating envelopes and performance ranges of the systems. The OZIX System Exerciser is used to load the system in a controlled manner. Subsystem interaction is stressed by running multiple copies of the same tests for various applications.

13.2.14 Configuration Testing

The Software Test Team is responsible for determining the minimum hardware configuration for OZIX. Testing of OZIX will be performed on some reasonable number of other configurations, some of which will include network devices.

13.2.15 Network Testing

A large amount of network testing is done in unit/integration testing and in testing applications which access the network. It is expected that TCP and DECnet certification will be performed by NaC.

13.2.16 Interoperability Testing

The interaction between OZIX and other systems (Workstations, VMS, ULTRIX) will be tested by the SVT lab. A separate interoperability test plan is being developed.

13.2.17 System Management Testing

There are two areas of concern for System Management testing. The first is testing the DECwindows interface. Some tools may be available in the future for DECwindows testing (see Section 13.2.5.4). Second, the current System Management design involves managing multiple machines. There are currently no tools to accomplish distributed testing.

13.2.18 Conformance/Compliance Testing

Conformance/compliance test suites are being run for POSIX 1003.1 and 1003.2, OSF, X/OPEN XPG3, ANSI C, Motif, Xlib and *NFS*. All of these test suites are being provided by outside organizations. It is expected that the development groups for SCS, MSCP, and LAT will obtain any internal conformance tests for these areas. The Software Test Team provides help in running of tests if needed. Network certification is discussed in Section 13.2.15.

13.2.19 Usability Testing

Usability testing is performed on components which are providing a user interface (System Management, On-line Diagnostic Monitor, System Exerciser, On-line Documentation, Messages). A separate usability plan is being developed.

13.2.20 B2 Certification

OZIX V1.0 will not be B2 certified, but should not preclude certification. OZIX V2.0 or some release between V1.0 and V2.0 will be certified. Testing for OZIX V1.0 takes security into account and is performed with the goal of having to duplicate as little as possible during B2 certification. The test plan may be refined as we obtain a better understanding of B2 certification.

13.2.21 Hardware Testing Strategy

The hardware testing strategy for systems running OZIX consists of several components, some of which are directly linked to the development of OZIX.

13.2.21.1 Hardware Testing Overview

It is expected that hardware systems on which OZIX runs are provided with a fairly extensive set of ROM-based diagnostic programs, a minimal set of standalone loadable diagnostic programs to cover any parts of the system boot path not covered by ROM-based diagnostics, and a relatively large set of online diagnostic programs.

The development of diagnostic programs is not a part of OZIX development. The following aspects of a hardware testing strategy *are* a part of the OZIX project:

- Development of the Online Diagnostic Monitor.
- Development of a system exerciser.
- Development of a system error logger and an error log report generator.

13.2.21.2 Online Diagnostic Monitor

The Online Diagnostic Monitor (ODM) provides the user interface and the controlling mechanisms for running online diagnostic programs. Online diagnostic programs are used to test or exercise system hardware and/or software without taking the system offline, thus causing a minimum amount of disruption to system availability.

13.2.21.2.1 ODM Architecture

The following are major highlights of ODM's architecture.

- It is designed to be independent of the hardware platform on which it runs.
- It provides a single, consistent, internationalized user interface for all ODM-compliant online diagnostic programs. Two versions of this interface are provided. One version is a DECwindows implementation, and the other is a command-line implementation capable of running on any type of user terminal.
- It provides a wide variety of user options, such as the ability to:
  - Run a single diagnostic program.
  - Run multiple diagnostic programs, either serially or in parallel.
  - Select the devices to be tested, the tests to be run on the devices, the length of time the tests should run, and whether to stop or continue when errors are reported.
  - Using default values for user options, simply select the devices to be tested and issue a "start" command.
  - Execute "script" buffers of ODM commands.

- It provides a set of runtime services to diagnostic programs for creating internationalized error report displays and user prompting messages.
- Users do not need to know diagnostic program filenames.
- Users do not need to know hardware device addresses.

13.2.21.2.2 Online Diagnostic Program Architecture

Following are architectural highlights for online diagnostic programs that run with ODM:

- They can be ported to all OSF-compliant environments since they perform I/O operations using X/OPEN-defined system calls.
- They can make use of diagnostic functions provided within OZIX device drivers, if non-portability is acceptable for selected diagnostics.
- They perform user terminal I/O via ODM services (which use underlying POSIX-defined system calls).
- They are internationalizable.

13.2.21.3 System Exerciser

The On-Line System Exerciser (OX) is a system exerciser that applies software loads to the system and exercises the hardware/software interface. This exerciser runs under the On-Line Diagnostic Monitor (ODM). ODM provides a standard interface to all hardware diagnostics including OX.

OX is utilized:

- By the network, I/O, and file systems groups to exercise their hardware/software interfaces
- By the performance group to apply specific loads to the system while taking performance measurements
- As one piece of the acceptance test for the system at customer installation

The use of OX provides:

- Exercisers for disk I/O, file system, network I/O, CPU, memory management, and tape I/O
- A method by which the I/O, network, file system, and performance groups can load the system in a controllable manner for their own testing
- A way to stress test the OZIX system software and hardware during development to aid in discovering software errors
- A tool for manufacturing to use to perform Stage III checkout and burn-in testing
- A tool for Customer Services to use to verify the installation of new hardware or software at the customer site and to troubleshoot system problems

13.3 Overall Quality and Testing Strategy

When implemented, this strategy demonstrates that the software product reliability and quality goals have been met. The overall strategy is to:

- Conduct design and code reviews
- Define the test plan
- Specify the component tests
- Plan for resources, including manpower and hardware and software
- Develop and document the tools and test suites needed to support the test plan
- Execute the test plans
- Feed back results as appropriate
- Track bug reports and fixes, improving test suites as needed and as resources permit

This strategy is a repetitive, adaptive process which continues over the life of the product set.

Many of the results of the test plan execution are data that can be used for both engineering and to form the basis for marketing documentation.

13.4 Risks

- Internationalization—The impact of internationalization and multi-language support on testing is unknown at this point.

13.5 Associated Documents

1. The *OZIX Software Development Procedures* which describes the process of developing components of OZIX.
2. The *OZIX Naming Standards and C Coding Conventions* which describes the project coding standards.
3. The *OZIX Performance Engineering Overview* which describes the performance engineering work for OZIX.
4. The *OZIX Security Certification Plan* which describes the strategy to obtain a B2 security rating.
5. The *OZIX Usability Plan* which describes the strategy for usability testing of OZIX.
6. The *Interoperability Test Plan* which describes the strategy for interoperability testing of OZIX with other systems.

GLOSSARY OF TERMS

***-property:** The property that a higher clearance level subject may not write to a lower sensitivity level object.

7x24 Operation: This means non-stop operation through 7 days of a week, and 24 hours of a day.

Access controls: Algorithms, architectures or hardware that control the access of subjects to objects.

Access stacks: A sequential set of modules that limit the access rights of subjects to objects. The access validation in OZIX is composed of one or more access stacks.

Access validation component: The OZIX component responsible for making access decisions.

ACSE: See *Association Control Service Element.*

Active State: A state through which objects are actively transitioning, i.e., arriving and departing

Address Resolution Protocol: The Internet protocol used to dynamically bind a high level Internet address to a low level hardware address

Agent: Provides operation dispatching services to manageable objects. At the core services layer, all management applications request operations to manageable objects via the agent.

AIA: Application Integration Architecture

Analytic Model: A software program that solves a series of explicit mathematical equations. Many queuing models are formulated as mathematical equations which are then solved. Often termed Back-of-the-Envelope calculations.

API: See *Application Programming Interface.*

Application Programming Interface (API): A layer of software that provides a standard-conforming programming interface for use by applications.

ARP: See *Address Resolution Protocol.*

Association Control Service Element: That portion of the OSI Application Layer common to all OSI Applications. It controls extended connections (e.g., *associations*) between OSI applications.

Assurance of Resources (AOR): An OZIX feature which assures that one or more users cannot prevent users of higher priority access to the resources of the system.

Attribute-based allocation (ABA): A storage management architecture, which uses.....

Authentication: The process of verifying the identity of a user. Authentication is especially important in a network environment where a message must be shown to have been issued by a particular user.

Authentication: A method of proving the identity of a user.

Availability: The probability that a system will function correctly at an instantaneous point in time.

Available: A level of service that withstands a single failure in each type of hardware component through redundancy, and restores service quickly in the event of software failures through fast reboot.

B2 Certification.: A level of security defined by the Department of Defense DOD characterized by structured protection

Backbone network: The collection of subnetworks responsible for forwarding data between other interconnected subnetworks.

Base containers: This is a container, whose physical configuration table enumerates only a single media in its entirety.

Basic Block: A region of the *CUE* that can be entered only at the beginning and exited only at the end

Black box testing: A method of software testing in which the tester is unconcerned about the internal behavior and structure of the program. Test data are derived solely from the specifications.

Block device: A random-access mass-storage device that conducts I/O operations using blocks or chunks of data.

Block device interface: Through this interface, data is transferred in multiples of sectors. Data transfer occurs between the media and the system buffer. Block device interface is accessible directly through the appropriate device special files.

BOOTP: A Internet protocol specified by RFC1084. This protocol is used by systems during their boot phase to obtain information about their environment. This protocol is not used by any vendors today and is currently inadequate for diskless workstations.

Boundary value analysis: A method of selecting test input and output data based on using values at the edges of equivalence classes.

Broadcast: A packet delivery system that delivers a copy of a given packet to all hosts attached to it.

Character device: A device that provides either a character-stream oriented I/O interface or, alternatively, a raw interface.

Character or Raw device interface: Through this interface, data can be accessed directly from the device. Data transfer occurs directly between the media and the user buffer. This interface is typically used by applications that have intimate knowledge of the data structure on the disk or tape device.

Clearance level: The part of a subject's label that determines what sensitivity levels of data he may access

Client: As part of a client-server model, the client is the system that initiates communications and requests service from a server. In the NFS system, the client is the component that accepts requests from users for file access and sends those requests to a file server. See also *server*.

CMIP: See *Common Management Information Protocol.*

CMOT: A double acronym for CMIP (Common Management Information Protocol) over TCP/IP. An Internet standard to support the CMIP protocol over TCP/IP connections.

Collector: Collector: collect and filter data. A collector gathers the gauged data from a set of probes or other collectors. It may then filter/process/archive them and pass them to other collectors or presenters

Common Management Information Protocol (CMIP): A protocol used for distributed network management. There are two flavors within DECnet/OSI, the DNA version, based on a draft of the second, the OSI standard for network management. (See also CMOT.)

Common Management Information Services (CMIS): The services needed to support CMIP.

Compartments: A grouping of related data.

Component Contribution Ratio: A high level representation of the sensitivity of overall system performance to a component's execution

Compound container: Container whose configuration table enumerates other compound containers, subcontainers or base containers. All the elements in a compound container are included in their entirety within the compound container. The components that are included to form the compound container are called constituent containers.

Confidentiality: Protecting data from unauthorized disclosure.

Connected endpoint (for datagram services): An endpoint that has pre-specified its destination port and address. Once set, the address cannot be changed for the life of the endpoint.

Container: This is a virtual storage device. It is uniquely named, persistent, and provides an array of data blocks for read and write operations. A container consists of container metadata and container data.

Core Services: The client/server orient services provided by the OZIX management backplane. Client applications can use the services provided by the agent, event dispatcher, event sink, and MIR servers.

Coverage analysis: A method of determining which lines of code were executed during a specific test.

Covert storage channels: An unusual method of passing information that violates the system's security policy

CTERM: A virtual terminal protocol specific to DECnet/OSI.

CUE: Component Under Evaluation is the collection of software that has been instrumented for performance characterization

DAP: See *Data Access Protocol.*

DASS: See *Digital Authentication Security Service.*

Data Access Protocol: The DNA Access Protocol. It is a proprietary remote file access service within DECnet/OSI.

Dataspace: A dataspace is a logical unit of storage used by higher levels of software, such as the file system or a data base application, to read or write data. Each dataspace has a unique, invariant name

Dataspace archive: The storage management space used solely for the purpose of archiving dataspaces. Storage manager migrates inactive dataspaces from dataspace migration domain into dataspace archive.

DCP: See *DECnet Copy Program.*

DECnet Copy Program: The application on OZIX and ULTRIX that implements the client side of the DECnet/OSI DAP protocol.

DECnet/OSI: That part of OSI embraced by the applicable DNA standards, plus extensions specific to DNA.

Digital Network Architecture: The collection of standards and architectures that comprise the DECnet specification.

Digital Time Synchronization Service (DTSS): A Digital product that provides distributed time services.

Discretionary Access Control (DAC): Access control based on the identity of the accessing subject

Discretionary model: The algorithm which defines DAC

Distributed Authentication Security Service (DASS): A Digital product that provides distributed authentication services based on RSA public key encryption techniques.

Distributed Name Service (DNS): A Digital product that provides distributed naming services.

Distributed System Security Architecture (DSSA): An architecture under development within Digital to provide strong security within a distributed system.

Distribution Services: The lowest layer of the management backplane. The distribution services insulate the high layers from the specifics of the management protocol and transport services used.

dlogin: The application on OZIX and ULTRIX that implements the client side of the DECnet/OSI CTERM protocol.

DNA: See *Digital Network Architecture.*

DNS: See *Distributed Name Service.*

DSSA: See *Distributed System Security Architecture.*

DTM: DEC/Test Manager

DTSS: See *Digital Time Synchronization Service.*

Elapsed time analysis: Analysis of a *CUE* that reflects the effects of the system (such as paging) on that particular run.

EMA: Enterprise management architecture. This is the architecture that describes system management of enterprise environment.

End node: A node within a DECnet/OSI network that does not involve itself with routing decisions.

Enforcement: Compelling obedience

Equilibrium Kinetics: The study of state transition kinetics - A classical methodology designed to study the dynamics of distributed systems. Found commonly in network performance studies

Equivalence partitioning: A method of partitioning of test input data into classes such that a test using a representative value of a class is equivalent to a test using any other value in the class.

Error: Occurs when a portion of the system assumes an undesirable state. If the error is observed at the output of the system by the user, it is considered a failure.

Event: A report that describes an asynchronous state change.

Event detection: The process of detecting an event. This is commonly referred to as tracing

Event Dispatcher: A server at the core services layer that provides manageable objects with a mechanism to post event.

Event Filter: A definition of the type of event to which the event subscriber listens.

Event Sink: A depot of events to be read by subscribers of events. Each event sink receives the event type defined by its event filter.

Execution context: The logical address space which can be referenced by an executor

Executor: A family of threads with the same security and accounting identities sharing the same set of SECs.

Executor model: The architectural model that defines an executor.

Exported filesystem: An entire local filesystem, a directory tree, or a single file that has been exported by a file server. An exported filesystem may be mounted by an NFS client.

Extended Services: Currently the highest layer of the management backplane architecture. These services are to be purely object-oriented and are TBD.

Failure: Occurs when the delivered service deviates form the specified service

FAL: See *File Access Listener*.

Fault: The underlying cause of an error, not necessarily physically identified

File Access Listener (FAL): The application on OZIX and ULTRIX that implements the server size of the DECnet/OSI DAP protocol.

Filesystem: An implementation defined disjoint tree of directories. A filesystem must be mounted into the system namespace before it can be accessed. See *namespace*. This is different from *file system*.

File system: The software that provides file services.

File Transfer and Access Management: The service and protocols defined by ISO for remote file access.

Fragmentation: The technique used by IP to break an outbound transport data request into segments that can be handled by the subnet-dependent device used to transmit the data. An identification number is placed in fragment for the destination to use in reassembling the datagram.

FTAM: See *File Transfer and Access Management.*

Gate: The linkage mechanism used to move from an SEC in one subsystem to an SEC in another subsystem.

Gateway: A computer that attaches to two or more networks and routes packets from one to the other.

Hooks: Involves the insertion of a trap code, jump instruction or some other type of dispatch mechanism. The process of enabling detection in this manner is commonly referred to as hooking

Identity stacks: a structure which records the execution history of a thread

Instrumentation: The process of adding additional code at various points in a CUE to record information

Instrumentation services: The infrastructure and interfaces available for obtaining instrumented data

Integrated Management: The same management framework for system, network, and application management in a distributed environment.

Integrity: The correctness of a program or of the state of the system; the protection against destruction or corruption of data

Integrity Access Control (IAC): Controlling access based on the amount of trust placed on code.

Integrity Level: A measure of how much faith one has in the correctness of a piece of code, the value of a piece of data.

International capabilities: The functions of an international product that support the languages and conventions of more than one locale.

Internationalization: The process that includes the development of an international product and the localization of the international product for delivery into worldwide markets.

Internet address: The 32-bit address assigned to hosts that want to participate in the Internet using TCP/IP.

Internet Protocol: The Internet standard protocol that defines the Internet datagram as the unit of information passed across the Internet and provides the basis for the Internet connectionless, best-effort packet delivery service. The Internet Protocol suite is often referred to as TCP/IP.

IP: See *Internet Protocol.*

IP datagram: The basic unit of information passed across the Internet. An IP datagram is to the Internet as a hardware packet is to a physical network. It contains a source and destination address along with data.

ISO: International Standards Organization.

Kerberos: A distributed authentication service based on private key encryption techniques, from project Athena.

Label: A combination of a security level, an integrity level, security categories and integrity categories.

LAN: See *Local Area Network*.

LAT: See *Local Area Transport*.

Least privilege: Allowing an executor to have no capabilities beyond those required to perform his duties

Level: Sometimes used as a synonym for label (see *label*). Also may refer to part of a label, either the security level or integrity level.

LM segment: An abstract vector of memory pages

Local area network: A physical networking medium designed for use within a limited geographical area. This term is frequently used to represent a class of high-bandwidth broadcast datagram networks, such as Ethernet.

Local Area Transport (LAT): A transport service protocol based directly on top of Ethernet, commonly used for terminal-host connections.

Locale: The local environment in which a product is used, including language, dialect, keyboard, data input and display conventions, collating sequence, and other attributes that directly affect how users interact with the product.

Localizable software: Software that can be modified to suit particular locales.

Localization: The process of adapting an international product to suit the language, conventions, and market requirements of a particular locale.

Location Broker: A form of name service specific to RPC, used to locate a specific remote service.

Logical memory segment: See *LM segment*

Maintenance Operations Protocol (MOP): A special low-layer protocol within DNA used for maintenance operations such as booting and diagnostic services.

Malicious: intending to cause harm

Manageable Object: An object that registers its class definition with the management backplane. Management operations performed on the object are issued from a management application, through the management backplane, and to the object.

Management Application: Applications that request the services of the management backplane to monitor and/or control manageable objects.

Management attributes: Part of the Enterprise Management Architecture (EMA) model. A named characteristic of a management object that may be changed.

Management Control Language: A semi-extension of NCL.

Management Information Repository (MIR): Contains class definitions of all manageable objects registered with the management backplane.

Management object: Part of the Enterprise Management Architecture (EMA) model. Each manageable entity defines what its objects are. An object is the target of the following operations: create, get, set, action and delete.

Management Service RTL: The run-time library (RTL) that management applications use to access the services of the management backplane.

Management Services: The set of services provided by the management backplane. This includes the services provided by the Extended, Core, and Distribution Services.

Mandatory Access Control (MAC): Controlling access based on the sensitivity level of data and the clearance level of users

MCL: See *Management Control Language*.

Message: Short descriptive text associated with a condition value.

Migration domain: This represents a set of containers. The storage manager uses the space to migrate active dataspaces, to enact storage management policies.

Model: Any abstraction of the real world. With reference to computer systems the term *model* is usually applied to a description of a computer system, component, or software program. Such descriptions are often codified into software programs that mimic, or otherwise behave as, that which they describe.

MOP: See *Maintenance Operations Protocol*.

Mount point: An empty directory on which another filesystem is attached.

Multicast: A technique that allows copies of a single packet to be passed to a selected subset of all possible destinations. Some hardware supports multicast by allowing a network interface to belong to one or more multicast groups. *Broadcast* is a special form of multicast in which the subset of machines to receive a copy of a packet consists of the entire set.

Multicast groups: A group of hosts that are logically grouped together. They share the same IP multicast address. Each host in the group receives a copy of a multicast datagram addressed to the group.

Multi-level devices: Devices which can contain data of more than one sensitivity level

Multilingual software: Software capable of supporting user interfaces in more than one natural language at a time.

Mutexes: A type of lock which assures MUTual EXclusion

Name service: A service that keeps track of names of objects known within a network, as well as attributes associated with each name, such as location or address.

National Computer Security Center (NCSC): The organization responsible for certifying the security of systems

NCL: See *Network Control Language.*

Need to know: The principle which states that a person should have access to only that data required to perform his job

Network Control Language (NCL): The application and its interface specification used to implement the client side of the DNA CMIP protocol on some Digital operating systems.

Network Interface Definition Language (NIDL): The specification language for Digital's RPC interface compiler.

Network layer: The third layer of the OSI Reference Model, responsible for relaying information between the source and destination transport entities.

NFS: Network File System

NIDL: See *Network Interface Definition Language.*

Node: A logical data origination or termination point within a network, usually associated with a physical computer system.

Nub: The most basic portion of the OZIX system which provides subsystems with primitive functions.

Object: Defined by its class definition (class name, attributes, operations, and events) and realized by creating new instances of its class.

Object Class: Definition of the object, which includes the definition of attributes, operations, and events.

Object Instance: The actual object itself. It is the instantiation of the object class.

Octet: Eight bits.

Open system: A system who's services and interfaces are known and available to outside parties.

Open Systems Interconnection (OSI): The term Open Systems Interconnection (OSI) qualifies standards for the exchange of information among systems that are *open* to one another for this purpose by virtue of their mutual use of the applicable standards. (ISO 7498)

Orange Book: Department of Defense Standard DoD 5200.28.STD, Dec 1985: the DoD Trusted Computer System Evaluation Criteria—the document that defines the requirements of the certified classes.

OSF: Open Systems Foundation

OSG: Open Systems Group

OSI: See *Open Systems Interconnection.*

Package: A set of subsystem procedures

Partitioned model: A model for replicating services to withstand failures. In this model some of the available resources act as primary providers of the service, and others act as passive secondaries.

Phase V DECnet/OSI: DNA has been designed in phases. Phase V represents the current architecture in which OSI protocols have been embraced.

Physical memory management: The system's management of its physical memory resources

Portal: A software mechanism providing a path of communication between communicating peers through a network based on incompatible protocols.

Port number: The number used to represent an address at the Internet transport level.

POSIX: Portable Operating System Interface for Computer Environments. A collection of IEEE standards.

Presentation: The sixth layer of the OSI reference model, responsible for negotiation and transformation of syntax between cooperating applications.

Priority: An assigned measure of the relative importance of a task

Profiling: The process of collecting and analyzing data that describes the run-time behavior of a *CUE*.

Proxy ARP: The technique used to allow one machine, usually a gateway, to answer ARP requests intended for another by supplying its own physical address. By pretending to be another machine, the gateway accepts responsibility for routing packets to it.

Rainbow books: The set of DoD documents issued by the National Computer Security Center, which defines the requirements and interpretations of certified computer systems. This collection of documents includes the Orange Book.

RARP: See *Reverse Address Resolution Protocol.*

Raw-device interface : The character-device interface for block-oriented devices such as disks and tapes. Through this interface, data transfer occurs directly to and from user's data buffers.

Reliability: The probability that a system will function correctly over a specified period of time

Reliability level: A measure of how much faith one has in the correctness of a piece of code, the value of a piece of data

Remote Procedure Call (RPC): In general, a style of network communication that is modeled after procedure calls. Clients make requests of servers that return replies. A particular implementation of RPC called Sun RPC is used by NFS for its communication.

Resources: Available source of supply or support that can be drawn on when needed.

Reverse Address Resolution Protocol (RARP): The Internet protocol that a diskless machine uses at startup to find its Internet address. The machine broadcasts a request that contains its physical hardware address and a server responds by sending the machine its Internet address.

Ring brackets: A two element value which controls the level of trust required to read or write an object

Ring levels: The level of trust of executable code

Robustness: Insensitivity to incorrect operation, inputs or errors.

Routing: The forwarding of data from the source to the destination along a series of one or more intermediate paths.

RPC: See *Remote Procedure Call.*

Sampling: Sampling is the process of examining event counters at some defined interval

SEC: The virtual memory accessible to an executor in a subsystem.

Secure RPC: A version of Sun RPC that uses the Data Encryption Standard (DES) for authentication. See *authentication.*

Security: Preventing the unauthorized disclosure of data

Security classes: Sets of objects subject to particular access controls

Security kernel: A relatively small set of code which implements a system's security

Security model: The algorithm controlling access to data based on sensitivity levels

Security policy: The rules defining how subjects may access data based on sensitivity levels

Sensitivity Level: A measure of how important the confidentiality of a data item is

Server: In a client-server model, the server is the component that receives requests from clients, executes the requests, and returns any results. In the NFS system, the server is the component that executes file requests on behalf of clients. See *client.*

Session: The fifth OSI service layer, defined by the Basic Reference Model to provide a common means of synchronizing the transfer of data.

Session control: A layer specific to DECnet/OSI that integrates the selection of protocols and use of underlying layers into the operating system on which it is running.

Simple Network Management Protocol (SNMP): An Internet-defined management protocol.

Simple security policy: The property that a subject may not read data which is more sensitive than his own clearance level

Simulation Model: A software program in which a computer system's hardware and software designs have been highly abstracted. Simulation models are used to answer performance questions that can not be answered with analytic techniques.

Smart card: An extremely small, self-contained processor used for authentication

SNA: See *System Network Architecture.*

Software Probe: The code that gets control upon the detection of an event. Probe: meter/observe events and status. By putting gauges in proper places and activating them allows the metering of events or status

SPC: See *Subsystem Procedure Call*

State equilibrium: Two cooperative states are said to be in state equilibrium when they are at

State Transition Rate: The change in the number of objects in a given state over some arbitrary interval of time

Steady State: A steady state is achieved when an active state's state transition rate equals to zero

Subcontainers: This represents a partition of a container.

Subjects: Active entities, generally in the form of a person, process, or device that causes information to flow among objects or changes the system state

Subnetwork: A topologically connected set of network entities sharing a common addressing technique. For example, an Ethernet or X.25 component of a larger network.

Subsystem: A specially designated body of code in the OZIX system. Code and data in one subsystem are protected from code and data in other subsystems by memory management. Multiple executors in the same subsystem are separated from one another and cannot communicate.

Subsystem Execution Context (SEC): See *SEC*

Subsystem Procedure Call (SPC): A mechanism for passing execution flow from one subsystem to another in a controlled manner. SPCs pass through security gates (see *gate*).

System Network Architecture (SNA): IBM's collection of networking architectures.

System State: System state information provides the state transition information. Some examples might include waiting on IO completion, executing, waiting on a lock

TCB: See *Trusted Computing Base*.

TCP: See *Transmission Control Protocol*.

Thread: A flow of execution

Time service: A service that provides accurate time and date information within a subnetwork.

Timing Analysis: The process of providing timing information, elapsed and cpu for the duration of an event. An event can be the execution of a particular command, program or a *CUE*.

TP4: See *Transport Protocol 4*.

Translation: The process of rendering information presented in one natural language into another natural language.

Transmission Control Protocol (TCP): A protocol in the Internet protocol suite. This protocol provides a reliable, virtual-circuit transport service.

Transport: The fourth OSI service layer, defined by the Basic Reference Model as providing transparent transfer of data between peer entities which relieves them from any concern with the detailed way in which reliable and cost effective transfer of data is achieved.

Transport Protocol 4: An implementation of ISO transport limited to operating at the fourth class of service as defined within the ISO Transport Service specification. An informal term.

Trojan Horse: A program that surreptitiously performs some set of actions while performing an alleged primary action.

Trusted Access Control (TAC): Access control based on the trustworthiness of code. In OZIX, this is based on ring brackets

Trusted Computing Base (TCB): The portion of the system software that is trusted to properly handle data at multiple security levels at the same time.

Unconnected endpoint (for datagram services): An endpoint that specifies its destination port and address for every transmitted datagram.

Upper layers: The common term used to reference the OSI Session, Presentation, and Application service layers.

User Presentation: A management application with user presentation services, which is visible to system administrators.

UTE: ULTRIX Test Executive

Vendor Assistance Program (VAP): A service provided by the National Computer Security Center to assist system developers in producing certifiable systems

Virtual memory model: The model that defines the relationship of an SEC to physical memory

Virtual Terminal Protocol (VTP): The service and protocols defined by ISO for remote terminal emulation.

VTP: See *Virtual Terminal Protocol*.

WAN: See *Wide Area Network*.

White box testing: A method of software testing in which the tester derives test data from an examination of the program's logic.

Wide area network: One or more physical networking media designed for use over an extended geographical area. This term is frequently used to represent communications over leased telephone lines or X.25 networks.

X.25: A CCITT recommendation defining services and interfaces to public packet switching networks.

X.400: A CCITT recommendation for services and interfaces between electronic mail transfer agents.

X.500: A CCITT recommendation for interfaces and services to a directory service.

X/Open Transport Interface (XTI): A transport service interface that is independent of any specific transport provider. It is defined by the X/Open Group.

XTI: See *X/Open Transport Interface*.

digital

The Phase 1 exit meeting of the OZIX Program will be held on Friday, November 17, 1989 from 1:00 to 4:00 p.m. at the Spitbrook facility, Cauchy conference room, ZKO3-1.

Enclosed are the OZIX Phase 1 documents. We ask that you carefully review these documents because the Phase 1 presentations will be summaries of these documents.

OZIX Phase 1 Agenda:

Opening Statement	*John Gilbert*
OZIX Vision	*Mike Parker*
Business Plan	*Terry Morris/Salley Anderson Teague*
Marketing Plan	*Mike Parker/Kevin Breunig*
Manufacturing Plan	*Rick Seed*
CSSE Plan	*LeeAnn Stivers*
Development Plan	*Benn Schreiber*
Documentation Plan	*Jim Jackson*
Phase 1 Closure	*John Gilbert*

There will also be a set of technical presentations on Friday, November 17, 1989 from 9:00 a.m. to 12:00 noon at the Spitbrook facility, Cauchy conference room, ZKO3-1. Each presentation will be approximately 30 minutes.

Technical Presentation Agenda:

Base System Architecture	*Chris Saether*
Security	*Jim Schirmer*
Internationalization	*Claire Cockcroft*
File System/Mass Storage	*Chris Saether*
Networks	*Steve Jenness*
System Management	*Mark Ozur*

The enclosed documents may also be found in a restricted distribution directory located at:

DECWET::OZIXV1$:[PHASE1]

The OZIX functional specifications may be copied from the following restricted distribution directory:

DECWET::OZIXV1$:[SPECS]

Each directory has a table of contents in the file AACONTENTS.TXT

Please note that all OZIX Phase 1 documents are Digital Confidential, with limited distribution based on absolute need-to-know.

If you have any questions, comments, or problems accessing the restricted distribution documents, please send mail to DECWET::OZIX.

DIGITAL
EQUIPMENT
CORPORATION

OZIX Vision Document

Revision/Update Information: 1.0

27 October 1989

Prepared By:

DECwest Engineering

digital™

Contributors:

Mike Parker, *OZIX Manager of Product Managers (DECWET::PARKER)*
Terry Morris, *OZIX Product Manager*
Debbie Walkowski, *OSG Documentation*
Kevin Breunig, *Software Systems Marketing*

Digital Equipment Corporation

Digital Equipment Corporation
Confidential and Proprietary

Trademarks of Digital Equipment Corporation

Concert Multithread	DECwindows	VAXcluster
DEC	Rdb/VMS	VAX DOCUMENT
DECnet	Rainbow	VMS
DECprint	ULTRIX	digital™
DECserver	VAX	

Other Trademarks

386 is a trademark of Intel Corporation
Ada is a trademark of the Department of Defense
AIX is a trademark of International Business Machines Corporation
Apollo is a registered trademark of Apollo Computer, Inc.
Apple is a trademark of Apple Computer, Inc.
IBM is a registered trademark of International Business Machines Corporation
INGRES is a trademark of Relational Technology Inc.
Macintosh is a registered trademark of Apple Computer, Inc
Microsoft is a registered trademark of Microsoft Corporation
MIPS is a trademark of MIPS Computer Systems Inc.
Motif is a trademark of Open Software Foundation, Inc.
MS-DOS is a registered trademark of Microsoft Corporation
MVS is a trademark of International Business Machines Corporation
Network Computing Kernel is a trademark of Apollo Computer, Inc.
Network Computing Software is a trademark of Apollo Computer, Inc.
NFS is a trademark of Sun Microsystems Inc.
Open Software Foundation is a trademark of Open Software Foundation, Inc.
OSF is a trademark of Open Software Foundation, Inc.
OSF/1 is a trademark of Open Software Foundation, Inc.
OSF/Motif is a trademark of Open Software Foundation, Inc.
POSIX is a trademark of the Institute of Electrical and Electronic Engineers Inc.
PostScript is a registered trademark of Adobe Systems Inc.
SUN is a trademark of Sun Microsystems Inc.
UNISYS is a trademark of UNISYS Corporation
UNIX is a registered trademark of American Telephone and Telegraph Corporation
X11 is a trademark of the Massachusetts Institute of Technology
X/Open is a trademark of the X/Open Group
X Window System is a trademark of the Massachusetts Institute of Technology

OZIX SYSTEMS PROJECT TEAM

OZIX Engineering Manager:

John M. Gilbert, *OZIX Engineering Manager*

OZIX Engineering Team:

Benn Schreiber, *OZIX Project Manager*
Marilyn Fries, *OZIX Development Supervisor*
Dick Funk, *OZIX Development Supervisor*
Mark Ozur, *OZIX Development Supervisor*
John Penney, *OZIX Development Supervisor*
Mike Peterson, *OZIX Development Supervisor*
Dave Snow, *OZIX Development Supervisor*
Bill Watson, *OZIX Development Supervisor*
Lu Anne Van de Pas, *OSG Compiler Development Supervisor*
Steve Jenness, *OZIX Network Architect*
Chris Saether, *OZIX ABA Architect*
Jim Schirmer, *OZIX Security Architect*
Claire Cockcroft, *OZIX Internationalization Program Manager*
Pete Benoit, *OZIX Project Leader*
Richard Brown, *OZIX Project Leader*
Sumanta Chatterjee, *OZIX Project Leader*
Jan D'Addamio, *OZIX Project Leader*
Mark Ditto, *OZIX Project Leader*
Dennis Doherty, *OZIX Project Leader*
Min-Chih Lu Earl, *OZIX Project Leader*
Debbie Girdler, *OZIX Project Leader*
Kelly Green, *OZIX Project Leader*
Jeff Havens, *OZIX Project Leader*
Brett Helsel, *OZIX Project Leader*
Oscar Newkerk, *OZIX Project Leader*
Charles Olivier, *OZIX Project Leader*
Kim Peterson, *OZIX Project Leader*
Jim Teague, *OZIX Project Leader*
Dave Walp, *OZIX Project Leader*
Charlie Wickham, *OZIX Project Leader*

OZIX Finance:

Judy Fox, *OZIX Business Support Manager*
Maggie Schimpf, *OZIX Financial Manager*
Salley Anderson-Teague, *OZIX Financial Analyst*

Product Management:

Mike Parker, *Manager OZIX Product Managers*
Kathy Appellof, *OZIX Transaction Services Product Manager*
Terry Morris, *OZIX Product Manager*

Cathie Richardson, *OZIX C & Third Party Applications Product Manager*

Manufacturing:

Rose Ramsey, *Software Manufacturing Product Manager*

Documentation:

Jim Jackson, *OZIX Publications Manager*
Liz Hunt, *OZIX Application Programming Environment Project Leader*
Marcia Aguero, *OZIX System Management Environment Project Leader*
Bill Talcott, *OZIX Software Support Environment Project Leader*
Cheryl Snyder, *OZIX Usability Engineering/Testing Project Leader*

Customer Services:

Bill Hilton, *OZIX Customer Services Systems Engineering Manager*
Thomas Siebold, *OZIX CSSE Maintainability Engineer*
LeeAnn Stivers, *OZIX CSSE Product Manager*
Myrna Harrison, *OZIX ESDP Project Leader*

TABLE OF CONTENTS

FIGURES

TABLES

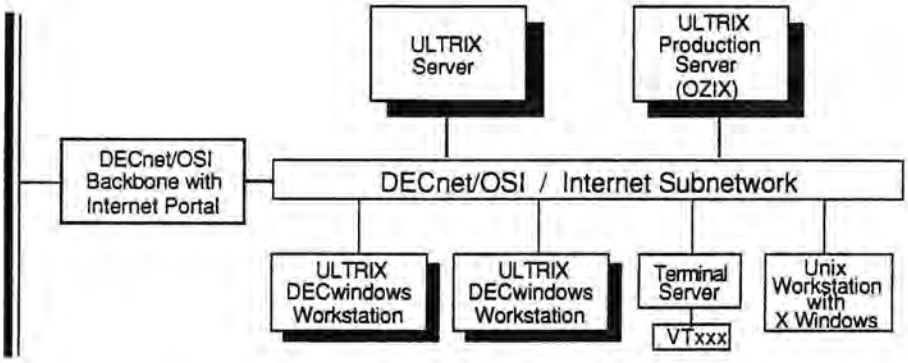
1 Introducing the OZIX Program and its Products

The OZIX [1] program delivers a series of a high-performance, reliable, open operating systems designed for production environments. The program provides the corporation with:

- An industry-leading operating system technology for the future
- Products that complete Digital's ULTRIX family of open, enterprise-wide computing systems
- Products that allow Digital to capture a large share of the growing open systems market
- Market leverage for Digital through introduction of technology into industry standards organizations

The mission of the OZIX program is to provide systems that allow Digital to compete in the growing open systems market, particularly where production systems are required. This is in keeping with the mission of the Open Software Group (OSG) to supply a complete family of open software products (see Figure 1) and with Digital's corporate mission to supply enterprise-wide information systems.

Figure 1: Typical 1990s ULTRIX Installation



OZIX systems provide solutions for a variety of customer needs in the UNIX marketplace. As *open* systems, they represent a superior implementation of open software standards. Targeted primarily at customers who require production systems for production environments, OZIX systems incorporate technologies that deliver maximum performance, fault tolerance, and reliability. They are optimized for superior availability and data integrity, providing a solid foundation for transaction processing (TP) and customized applications. In addition, OZIX systems serve as compute and file servers in environments that require enhanced security and reliability. As integral members of multivendor networks, OZIX systems interoperate with UNIX, IBM, VMS, and ULTRIX systems.

[1] *OZIX* is a code name for a program. For simplicity, the term OZIX is used in this document as a product name to refer to the series of products that complete the ULTRIX product family. Products produced by the OZIX program will ultimately be named consistently with ULTRIX products.

2 Why OZIX?

OZIX is designed to respond to needs in the UNIX marketplace where specific weaknesses exist today. Weaknesses in current UNIX products include:

- Minimum level of system security
- Lack of reliability and availability of applications and data (for example, in transaction processing systems)
- Restrictive file structure
- Inadequate software tools for the productivity of application developers
- Obscure user interface
- Unapproachable documentation that is incomplete

These weaknesses have created a new market, a "non-traditional" UNIX market, in which the requirements for UNIX systems have grown substantially.

3 The Non-Traditional UNIX Market

Customers who purchase UNIX systems today fall into three categories: tradition-driven, technology-driven, and policy-driven. The non-traditional UNIX market encompasses technology- and policy-driven customers.

- Tradition-driven—These customers are interested in public-domain source code and have little interest in a vendor's added value. These customers are typified by the education market; they want UNIX for the sake of UNIX.
- Technology-driven—These customers purchase solutions that happen to have UNIX-based software platforms. They seek access to new technology that offers leadership price/performance and third-party applications. Solutions purchased by these customers are used for many applications, including vector processing, ECAD, retail, and financial analysis.
- Policy-driven—These customers are primarily concerned with protecting their investment in data and applications. They balance the need for a stable application environment against the desire for new technology and innovation. Governments and their contractors have traditionally driven this market, but now many firms interested in protecting their applications investments are seeking UNIX solutions.

Dataquest projects the UNIX market will be 18%, or $48.5 billion ($24 billion for initial system sales), of the total electronic data processing (EDP) market by 1992. Of this market, technology- and policy-driven customers represent the largest share. While tradition-driven customers have built a substantial framework for UNIX by demanding standard application programming interfaces (APIs) and utilities, their overall system requirements have not changed or expanded significantly over the years. Thus, they represent only a small segment of the projected UNIX market.

Technology- and policy-driven customers demand much more than standard APIs and utilities; they demand improvements such as:

- Intrinsic reliability and availability of applications and data (for example, in transaction processing systems)
- Leadership performance
- Stringent security
- Lower operating costs with enterprise-wide management capability

- Software tools that improve the productivity of application developers
- Interoperability with multivendor, distributed systems
- A broad set of third-party applications
- Systems that can be used worldwide by users in their native language
- Approachable documentation that is easy to use
- Expanded service and maintenance facilities tailored for the production environment

Many of these improvements point to the need for "production" systems in the open systems marketplace.

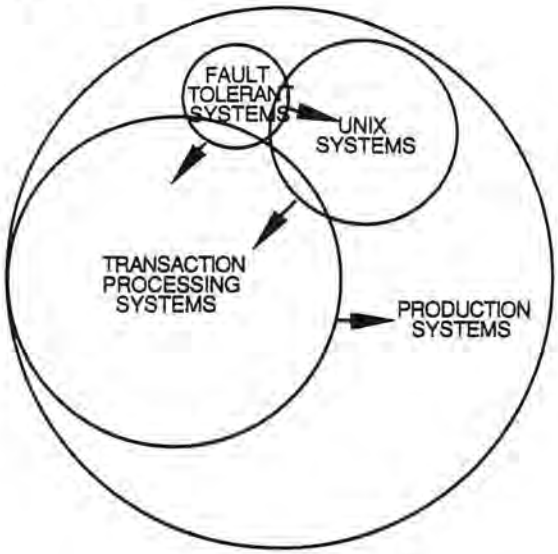
4 The Computing Environment: Production Systems

OZIX systems are targeted at the technology- and policy-driven customers whose computers operate in a "production" environment. A production system is characterized by a mature, tested application running continuously to support an ongoing (often real-time) process. The system is constantly tuned to balance the workload for optimum performance and immediate response time. Because of the continuous nature of the process, shutdown of a production system is extremely costly—down time directly impacts the customer's profitability. In a production computing environment, cost, system throughput, and consistency of response time are key concerns. The three key resources—the application, the database, and the system itself—are all managed by a dedicated operations staff, not the end user.

In contrast, in an end-user computing environment, applications are not critical, down time does not necessarily affect profitability, the system is not continuously tuned, and the end user is often able to manage the key resources.

The market for production systems is illustrated in Figure 2. The figure indicates that the production systems market encompasses fault-tolerant systems, UNIX systems, and transaction processing systems. Transaction processing systems are the dominant form of production computing today. These systems will continue to pervade the production systems market while UNIX and fault-tolerant systems will become greater requirements for TP systems.

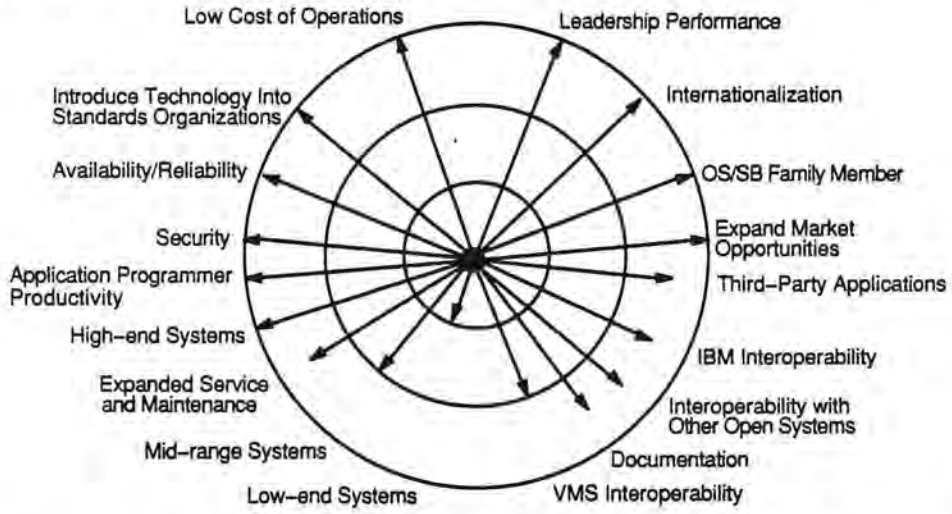
Figure 2: Production Computing Market



5 OZIX Development Strategy

Development priorities for OZIX directly reflect what is known about the production systems market and the improvements that technology- and policy-driven customers want in UNIX systems. These priorities translate into product features, which are illustrated in Figure 3.

Figure 3: Priority Star for OZIX Product Development



By incorporating these features into the base system of OZIX products, Digital is able to provide a solid solutions platform for a wide range of applications in open systems, production environments. Examples of such customer applications are listed below:

- An aerospace company connecting its CAD workstations to a secure data repository for large government projects

- A natural resources company reducing geological data that requires accurate answers and quick completion of a compute job that runs for several days
- A retail organization collecting data from point-of-sale terminals onto a highly available system to provide inventory analysis
- A telecommunications company running a high-performance transaction system for billing and network control applications
- An automotive company collecting work-in-progress information from a manufacturing system, then generating parts orders that are sent electronically to its vendor
- A financial company analyzing large amounts of data generated by financial markets
- An insurance company processing claims up-loaded from agents' systems or consulting an online actuarial system

Currently, Digital is limited in its ability to provide solutions in these application areas. OZIX products enable Digital to compete effectively in the open systems production marketplace.

6 OZIX Products Offer a Competitive Advantage

Customers buy products that allow them to establish and maintain an edge over their competition. Likewise, Digital seeks to develop products that give us that same edge over our competitors.

6.1 The Competitive Advantage—from the Customer's Perspective

OZIX products contribute to the customer's competitive advantage in the following ways:

- Low Cost of Operations

  By simplifying the tasks of managing systems, OZIX products allow a customer to reduce the size and expertise of their operational staff. From a central location, a staff of ten should be able to manage a system consisting of ten servers and 1000 workstations. Routine tasks such as printing, or mounting backup media, should be accomplished by non-technical personnel, further reducing management costs.

- Application and Data Availability

  By providing uninterrupted service,[2] an application running on an OZIX product is ready and capable of returning the stored data whenever the data is requested. Failure of the operating system does not interrupt service to the customer's application, but instead provides transparent failover. In the rare event that a system shutdown does occur, the system degrades gracefully, keeping its data safe and, if required, provides a fast system boot and application restart.

- Data Integrity

  OZIX products provide fast, flexible, and reliable access to data and ensure that data is safe from corruption. This is accomplished through the combination of available hardware, and reliable software that incorporates transaction processing features. Given these features, the transaction will either commit correctly and completely or will not commit at all.

- Investment Protection

[2] OZIX products are required to be shut down no more than once per year.

By incorporating industry standards, OZIX products protect the customer's investment, allowing easier yet secure access to their data. Customers can write applications once and move them anywhere throughout the computing environment. Users need not be retrained since interfaces are consistent across all of their open systems platforms. For the same reason, application programmers need not be retrained whenever they move to a new project. Data is easily used within the open systems environment because all data is formatted in a consistent fashion.

- Productivity of Application Programmers

  By integrating the best of Digital and third-party tools, OZIX products provide a CASE platform designed to reduce an application's time-to-market and produce applications that are easily maintained. OZIX products leverage the ULTRIX CASE platform by providing additional tools for the production systems environment. Programmer productivity is the focus, helping end users reduce their three- to four-year backlog in application development and improving third-party software vendors' product delivery.

- Application Portability

  By complying with X/Open's XPG4 [3] and OSF's AES level A, applications written for OZIX products can be easily ported to new hardware platforms. The added benefit is that these applications are highly portable within a robust environment.

- Integration into Multivendor Computing Environments

  By using interfaces such as TCP/IP, NFS, OSI, and RPC, OZIX products are able to interoperate with other open systems. By incorporating AIA components and DECnet/OSI, OZIX products interoperate with VMS systems. Support for SNA, LU6.2, and Digital's VIDA products allows interoperation with IBM systems as well. In addition, OZIX products offer an integrated ANSI SQL, which provides an industry-accepted method for accessing data.

- Security

  By complying with the National Computer Security Center's criteria for B2 level security, OZIX products demonstrate their strong resistance to intrusion and their protection of data from unwanted disclosure. OZIX products allow customers to tailor the security features to meet their specific requirements, from full Department of Defense (DoD) policy, to policies and enforcement levels more suited to a commercial environment.

- A Wide Range of Applications

  By building upon the applications currently available for the ULTRIX operating system, OZIX products expand the platform for open systems applications. For example, by employing complex system features such as concurrency and TP primitives, an automobile manufacturer is able to run a Material Requirements Planning (MRP) system.

- High Performance

  Through efficient implementation and a focus on I/O and network performance, OZIX products enable applications to perform better than on any other comparable open systems platform. To ensure high run-time performance, OZIX products provide self-tuning mechanisms that maintain optimum operating system performance.

- Systems That May be Used Worldwide

[3] It is expected that XPG4 (X/Open Portability Guide 4)will be available by FRS for OZIX.

By incorporating support for multilingual user interfaces and local cultural conventions, OZIX products allow multinational customers to place systems worldwide regardless of the end users' native language. OZIX products support multiple character sets, including those for major Asian languages. They are flexible, global software products that provide applications with the mechanisms allowing users to interact with the system using their choice of native language and cultural conventions.

6.2 The Competitive Advantage—from Digital's Perspective

OZIX products contribute to Digital's competitive advantage in the following ways:

- Attract Third-Party Applications

  By incorporating high-performance TP technologies into the base system, OZIX products attract a new set of applications needed by Digital's ULTRIX customers. In addition, a rich set of application development tools attracts not only customers but also third-parties. Digital benefits by having more third-party applications on the market earlier than its competitors.

- Capture a Large Share of the Open Systems Market

  By attracting a broad range of new applications, OZIX products enable Digital to capture the "newest" share of the open systems market, the policy-driven and technology-driven customers. OZIX products also provide the required open systems APIs within a stable development and run-time environment, further leveraging Digital's ability to capture a large share of the open systems market.

- Provide a Growth Path for our Customers

  By completing the high end of the ULTRIX product family, OZIX products allow Digital to offer customers a full range of open software systems. Customers want to store data and run the same applications across a wide range of platforms. As their applications grow, they want to rescale to larger systems. The ULTRIX product family leverages Digital's ability to provide customers a wide range of system performance without the need to modify their data or applications.

- Provide a Better Open Systems Environment

  By providing a superior implementation of open software standards, OZIX products provide Digital with product distinction. The OZIX product architecture incorporates transaction processing concepts, robustness, and security as core features. Existing UNIX implementations must be substantially reimplemented to incorporate these concepts into their existing architectures.

- Be an Open Systems Technology Leader

  By developing technologies that can be selectively submitted to standards organizations, OZIX provides Digital an avenue to develop open systems technologies based on standards and allows for Digital added value. Traditionally, UNIX vendors create de facto standards and hence technology leadership by making strategic technologies widely available. Domination in key industry technologies is leveraged through OZIX and other Digital technologies such as the Concert Multithread Architecture (CMA).

7 Positioning OZIX as a Product

Digital has two primary operating system families: VMS and ULTRIX. With the introduction of OZIX products, the ULTRIX family is expanded to meet emerging customer demands in non-traditional open systems markets. Because VMS and ULTRIX products target different markets and offer distinct capabilities, both are critical to Digital's success.

7.1 OZIX Products Within the ULTRIX Family

OZIX products are members of the ULTRIX family. They are aimed at markets where UNIX (and traditional ULTRIX) systems are not generally used today. OZIX products allow Digital to expand its opportunities in the open systems market by delivering high-performance, highly secure, distributed systems to an international marketplace. Because OZIX products are built with the latest design and engineering methodologies, these capabilities are fully integrated into the base system.

Digital's ULTRIX family spans a broad range of functionality from a low-cost implementation of OSF/1 to the high-functionality implementation of OZIX products.

7.2 OZIX Products vis-a-vis VMS

The VMS operating system is the most complete, functional operating system on the market today. It successfully addresses the needs of customers across a broad range of markets. The VMS operating system's primary competition is IBM's MVS and AS/400 production systems. It is important to understand that VMS and open systems customers have similar needs. The difference is that the open systems customer's primary requirement is for an open systems interface; production environment features are a secondary requirement. In contrast, VMS customers require high functionality, and open systems interfaces are not necessarily a requirement.

7.3 OZIX Products vis-a-vis Other Open Systems Offerings

Current open systems offerings will have serious problems providing customer solutions for tomorrow. Below are two excerpts from "The Architecture of Future Operating Systems," by Richard W. Watson, Lawrence Livermore National Laboratory.

- Bill Joy—"We see a lot of people having multiprocessors; we see a lot of people having distributed systems connecting lots of computers together, and that's not the kind of environment that there was when UNIX was first defined so *we want to redefine what UNIX is so that it really works* [italics added]."
- Professor Richard Rashid, Carnegie Mellon University—"UNIX no longer is easy to modify as it once was....B.S.D. contains more than 100 system calls and hundreds of system call options....the UNIX kernel has become a dumping ground for virtually every new feature or facility."

Mr. Watson lists the following as known problems with current UNIX implementations:

- Extension and customization requires kernel modification
- Lack of features that support a robust computing environment, for example, availability
- Known security holes
- Distributed computing deficiencies
- Modularity and implementation problems
- Weak support for high-performance mass storage and large-capacity files

OZIX products address each of these issues directly by providing a new modular implementation which permits the system, including the executive kernel, to be modified and extended by customers and third-party vendors to meet new and unique application requirements.

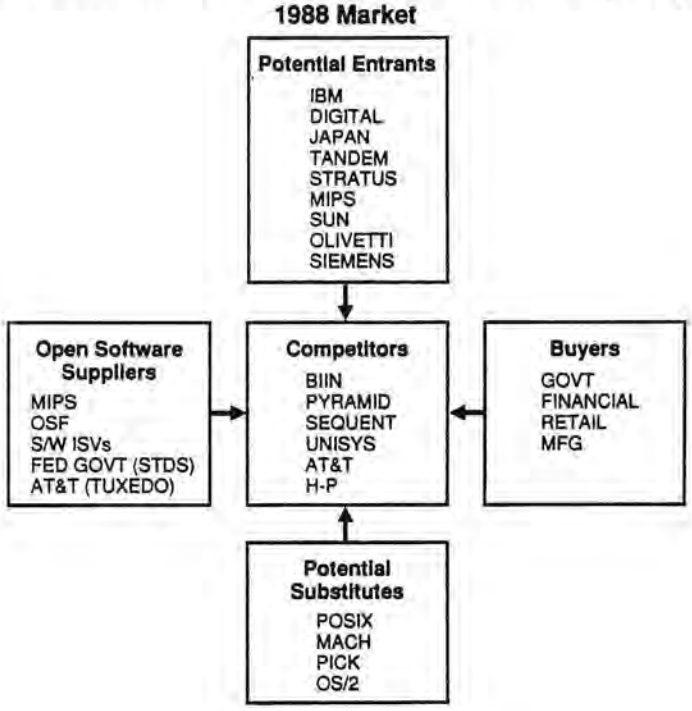
7.4 Competitive Environment

Figures 4 and 5 are evolved from Michael Porter's "Competitive Forces Model".[4] They depict the competitive environment in 1988 and project the competitive environment in 1992.

7.4.1 The 1988 Environment

The 1988 open production software environment is considered by many to be small and immature. This is due to the lack of transaction processing features in most current implementations of open systems. The model below suggests that the environment will change as vendors such as Hewlett-Packard, AT&T, UNISYS, and Pyramid invest their software engineering resources to protect and then expand their current base of open systems commercial customers.

Figure 4: Competitive Environment for Production Systems in the Open Systems Market

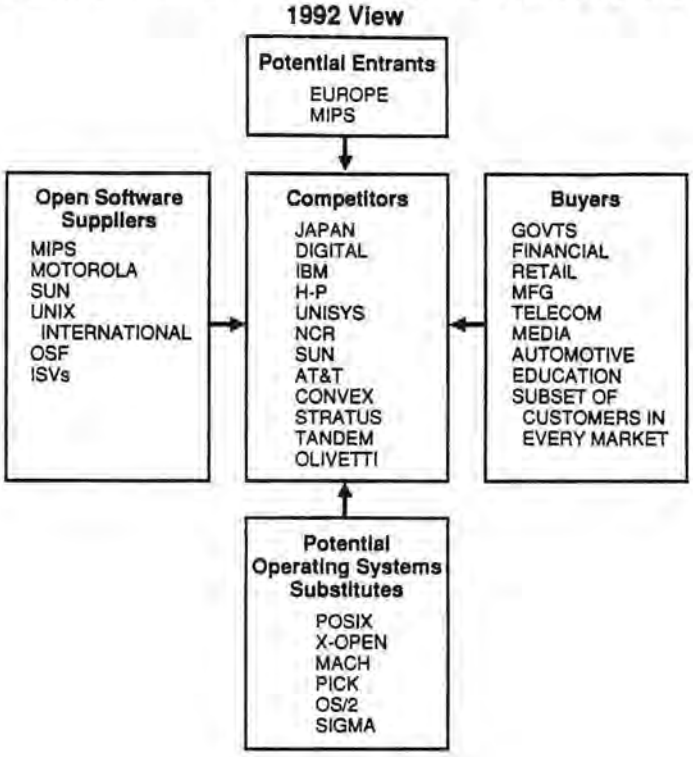


7.4.2 The 1992 View

As vendors rush to deliver solutions to customers who require open, production systems, the open systems market will grow worldwide. Our view of the worldwide market must truly be worldwide—if our view is only on North America, the activities of non-North American competitors will not be measured. There is great concern over Japanese manufacturers entering the open systems market. Many believe the Japanese vendors will temporarily pass up the United States market to exploit the faster growing European and Asian markets.

[4] From "How competitive forces shape strategy" Michael E. Porter, Harvard Business Review March-April 1979 pgs. 137-145

Figure 5: Competitive Environment for Production Systems in the Open Systems Market



8 OZIX Product Description

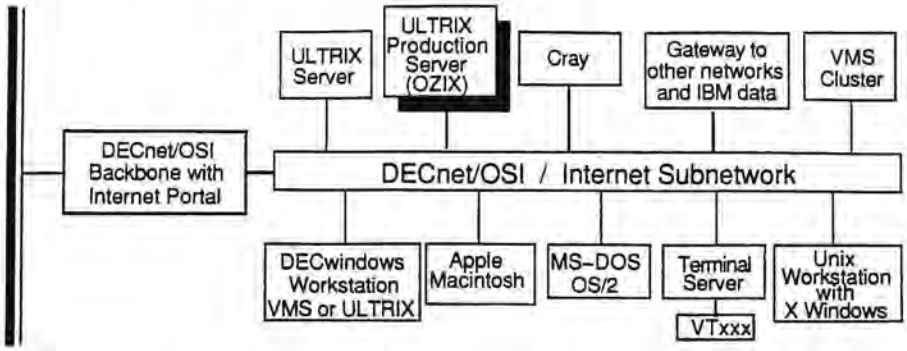
OZIX is a simple, extensible, hardware-independent operating system. It integrates modern technologies for distributed systems, fault tolerance, and data integrity, while adhering to a high-performance I/O architecture. Its capabilities are implemented over a series of releases and are composed of system elements (such as database and transaction processing software, compilers, tools, and utilities) from other Digital groups, as well as selected elements from third-party vendors.

Version 1 of the OZIX product is a high-performance production system targeted at the open systems market. While transaction processing technologies are built into Version 1, these capabilities are not accessible to the application developer until Version 2. In the first release of the product, the system will utilize these transaction technologies to prove its high level of performance and reliability. Proof of these capabilities establishes a successful track record for Digital in the open systems marketplace with a high-performance production system.

Version 1 of the OZIX product is presented as a high-performance open system designed to function as a network server in a multivendor, distributed systems network (see Figure 6). Version 1 serves this network by providing file services (via NFS), compute services (user applications), workstation services (such as workstation system management, booting, and diskless services), and database services.

Also included in Version 1 are integrated system and network management features, license management, and CDROM distribution. Portability of applications is provided with Application Integration Architecture (AIA) components such as DECwindows client, Compound Document Architecture (CDA), and Concert Multithread Architecture (CMA).

Figure 6: Digital's Computing Environment in the 1990s



Establishing a successful track record with Version 1 of the OZIX product allows Digital to address more demanding production system requirements with Version 2. The transaction processing technologies built into Version 1 become available to the application developer in Version 2. The goal for this release is to be the highest performance TP system both in terms of transactions per second per MIP, and in terms of dollars per transactions per second. In addition, Version 2 will provide world-class distributed system administration via a graphical user interface. Support for new AIA component architectures will continue to be expanded.

8.1 OZIX Product Features

- Fault Tolerance, Reliability, and Availability

  A system-wide architecture for error logging, fault detection, and recovery is integrated into the base system of OZIX products. This architecture works in conjunction with hardware such as fail-stop processors and Digital's Computer Interconnect (CI) to provide the highest degree of system and application availability.

- Robust, Recoverable File System

  The OZIX file system provides a high degree of availability and data integrity by exploiting transaction processing techniques to guarantee robustness and minimize recovery time. The file system also utilizes Attribute Based Allocation (ABA) to provide hierarchical storage management and storage migration.

  The OZIX file system supports the POSIX 1003.1 file system interface, with additional resource management services to support database systems in a production environment.

- Superior Implementation of Open Software Standards

  The OZIX program delivers a novel operating system in which the basic system concepts reflect the models of process management, file processing, and system calls defined in the OSF, X/Open, SVID, POSIX and OSI standards. By taking these key concepts as the "native" mode, OZIX products execute applications written to the standards more efficiently than conventional proprietary systems that are designed to different architectural models with standards interfaces as layers. OZIX products are superior to existing UNIX implementations because their architecture incorporates transaction processing concepts, such as robustness and security, as core features. Existing UNIX implementations must be substantially reimplemented to incorporate these concepts into their existing architectures.

  The combination of standard interfaces and integral production system features make OZIX products a unique offering in the open systems marketplace.

- Foundation for Building Innovative Applications

  OZIX products enhance the standard application environment by providing the ability to distribute applications in a local area network. Symmetric multiprocessing support, multithreading, and compiler-supported parallelism make it possible for applications to use multiprocessor concurrency. Remote procedure calls and a network file system support innovative distribution of applications across a local area network. These features also contribute to data integrity and high availability of applications.

  OZIX products provide an implementation of the Digital Distributed Transaction Architecture (DDTA), providing the basis for the development of highly reliable systems oriented for transaction processing applications.

- I/O Performance and Connectivity

  OZIX products strive to offer the highest mass storage and network throughput per unit of computational power, as well as the lowest latency time in access. The overriding concern in this design, however, is to preserve the reliability and integrity of the customer's data. Disk striping, connectivity, and advanced storage management contribute to superior I/O performance.

- Scalability

  The modular design of OZIX allows features and functions to be added without significantly increasing kernel size, and thus increasing system overhead. As ULTRIX workstation customers upgrade to larger systems, they are able to retain their operating system without incurring conversion costs.

- Security Model

  Because of the worldwide trend toward more secure systems, OZIX products are designed and built for certification at DoD security level B2. Customers can tailor the security features of OZIX products to meet specific installation and application requirements. In addition, by providing a flexible security model that allows customers to tune the system to their needs, customers do not pay a performance penalty for unwanted security features.

  OZIX products also support extendible security models, providing flexible security of user data. Distributed user identification and authentication mechanisms such as Athena's Kerberos, and in the future, DASS, are provided for customers who must operate worldwide networks and need enhanced distributed system services. As new distributed security protocols and authentication mechanisms become available, they can be added to the modular security model for OZIX.

- Integrity Model

  Recognizing that the protection of data from contamination and destruction is vitally important in the commercial environment, OZIX products provide pervasive integrity support and formal integrity models to control data access. Various integrity models can be supported, allowing the customer to match the integrity controls to their particular needs.

  Operating in conjunction with the OZIX security model, the integrity model protects the customer from inadvertent disclosure, modification, and destruction of valuable data.

- Superior Network Implementation

  The OZIX networking environment is designed to support systems from multiple vendors participating in both local and wide area networks. OZIX products employ a combination of DECnet/OSI and TCP/IP components. TCP/IP is the protocol for the local area network subnet; a DECnet/OSI backbone gateway with a TCP/IP portal provides wide area network access. DECnet/OSI also provides VMS interoperability. IBM SNA interoperability is provided through network gateways.

- Internationalization

  OZIX products are designed to be sold in international markets and, therefore, may be described as language-neutral, culturally unbiased products. Critical components such as the file system, application programming interfaces, and terminal services, have the built-in capability to accommodate worldwide character sets for textual representations. OZIX systems include an extended I/O package that handles a variety of text formats to perform frequently used text operations. These features ensure that customers and third-party developers are able to create applications tailored for specific cultures and local languages.

- Third-Party Applications

  Independent software vendors play a key role in the OZIX program. By offering a rich set of third-party software development tools and utilities, OZIX products reduce development time, thus providing a means for achieving and maintaining a leadership position in the open systems marketplace. The availability of a broad range of third-party application packages also increases Digital's opportunity to sell OZIX products by satisfying the customers' needs for complete *solutions*, not simply *systems*.

- System and Network Management for Production Environments

  OZIX products provide an advanced system and network management interface for easy management of a 24 hour/day, 7 days/week production environment. The goals are to maximize the system manager's productivity and lower the overall cost of system management. These goals are achieved by providing a window-based workstation user interface, an architecture that merges system and network management, and unified management of workstations and servers. This management environment is architecturally compatible with all other ULTRIX products and with Digital's Entity Management Architecture (EMA).

- Documentation

  The OZIX product information set is structured specifically for online presentation in a hypertext information style. Although hardcopy documentation is available, the primary retrieval method will be online.

  The OZIX product information set is tailorable for the needs of different types of users and is designed for easy translation to other languages. In addition, publications tools are being developed to integrate training modules and documentation from other sources (for example, OSF, third-party vendors, or Digital) into the OZIX product information set.

- Serviceability for Production Environments

  OZIX products are designed to provide uninterrupted service, 24 hours/day, 7 days/week. In no case should the system require more than one shutdown per year. Uninterrupted service is achieved through proactive, preventive methods of service based, in part, on comprehensive event monitoring, error handling, and automatic notification capabilities. In addition, software updates may be handled without system shutdown.

- Commercial Computing Requirements

  Two important aspects of commercial computing that OZIX products address are distributed printing and distributed batch processing. OZIX products execute batch jobs on the appropriate hardware automatically without user or administrator intervention, and operators are able to examine and modify the status of a job executing at any location. OZIX products provide print server capabilities that allow print jobs to be scheduled, restarted, directed to special-purpose printers, divided between two or more printers, and so on.

- Familiness

As an integral member of Digital's ULTRIX family, OZIX products share a common family heritage with all ULTRIX products. Each member of the ULTRIX family shares:

- A common set of application programming interfaces (APIs)
- A common set of application development tools, utilities, and commands
- A single network and distributed systems architecture
- A single system and network management architecture
- A common core of documentation

This compatible set of base systems provides customers and developers with network interoperability and a high degree of application portability across Digital hardware architectures.

- Relational Database Technology

  OZIX products support third-party relational databases as well as Digital's high-performance RdbStar. RdbStar is designed for enterprise-wide systems participating in multivendor distributed environments. RdbStar will capitalize on the designed-in data integrity of OZIX products to ensure 100% data integrity in production and TP application environments. In addition, RdbStar will incorporate industry standards, placing Digital in a leadership position in the database market.

9 OZIX Product Marketing Strategies

Production environment customers will only risk critical corporate data processing resources on established products. For example, a recent survey [5] reported that the criteria most frequently cited by Management Information Systems (MIS) directors as the "most important factors" for the selection of open production systems are:

- Backing of an established vendor
- Functionality of the UNIX implementation

In order to establish OZIX as a production system and to incorporate the correct functionality the OZIX program will make use of an advisory council and will carefully control early releases. The advisory council is made up of third-party developers and end users that need open production systems. Throughout the development cycle the council reviews OZIX's functionality, prioritizing the delivery of the functionality. OZIX's product entry point is focused on a limited set of applications, shipping into environments where product marketing, the field, and engineering have established close customer relationships. From this, OZIX evolves as the customer's needs evolve and as Digital gains the experience supporting open production systems. With each release of OZIX, more of its underlying technology is exposed to more developers and customers.

Below is a brief outline of the steps for bringing OZIX to market.

[5] Source: "MIS Considers UNIX: Opening Wallets and Opening Minds," IDC UNIX Service, May 1989

Table 1: Marketing Activities

Time	Phase	Activity	Risks/Dependencies
1989-92	Pre-Release	prioritize and identify key applications, identify key third-parties, identify key channels, establish an advisory council made up of developers and end users to communicate and check vision, start third-party porting activities, build "real-time" market data feed	Evolving product, Non-disclosure risks
1992-93	Trial	Help key third parties to port their applications. Help seed accounts port applications, generate success stories. Package hardware, software and service into turnkey solution to ease ordering. Generate cross-industry awareness	Need Co-operation of many PBUs, PMGs
1993-	Ramp-Up	Announce new third-party applications, Work with industry specific System Integrators/accounts in telecommunication, manufacturing, departmental general ledger	Need field support facilities

OZIX is targeted at the open systems medium-large commercial/TP market. Below is a table that profiles each open systems market segment. It describes the customer profile, the competition, the selling channels, summary product needs, communication media, and a summary pricing strategy.

Table 2: Market/Strategy

	Small-Medium General Business	Technical Workgroup	Medium-Large Commercial/TP
Customer	Price sensitive, interested in low cost, turnkey solutions for retail, manufacturing, distribution.	Sophisticated, interested in reliable server; tools important, applications less so	MIS organization wants highly available, open platform. Must access existing data. Mandated UNIX operating system or investing in new low cost equipment. Innovator in a risk averse environment.
Competition	NCR, Unisys, H-P, IBM, Olivetti	SUN, H-P	IBM, Stratus, Tandem, Pyramid, AT&T, Unisys
Channels	Value Added Resellers	Value Added Resellers, Mailorder, Direct Channels, Direct	

Table 2 (Cont.): Market/Strategy

	Small-Medium General Business	Technical Workgroup	Medium-Large Commercial/TP
Product	Low cost, distribution solutions with bundled applications., complete solutions, price growth path	Standards, interoperability, security, price performance	Standards, Vendor reputation, IBM interoperability, price performance
Communications	"Dealerscope", "Merchandising", "ABA Journal"	"Unix Review"	"Datamation", "Computerworld"
Price	Low entry price, limited options	Many options, configurations, industry standard busses	Trade-in allowances, free installation, etc.

10 Additional Information on OZIX Products

This document has presented a brief introduction to the program and vision for OZIX products. More detailed information regarding OZIX architecture and design may be found in a variety of phase review documents, including:

- **OZIX Market and Product Requirements Document**. This document defines and describes open systems, summarizes customer requirements for production systems in the open systems market, and provides an analysis of the market opportunity. OZIX product requirements are presented relative to these findings.
- **OZIX Product Features Document**. Specific OZIX product features and their implementations are described in this document. The eight major categories of features include versatility, standards compliance, commercial features, application environment, operations environment, system stability, performance, and distributed computing.
- **A Vision of the International OZIX Product**. This document outlines the goals and strategy for the internationalization of OZIX products. OZIX products are engineered from their inception for use in all Digital markets worldwide and are designed for simultaneous worldwide product release.
- **OZIX Applications Plan**. Third-party application support plays a major role in the OZIX program. This document examines the target markets for OZIX products and the types of applications that are needed to ensure success in these markets. It addresses the issues of training, documentation, and internationalization for third-party applications, and the program for procurement of these packages is outlined in detail.
- **OZIX Base System Architecture**. This paper describes the architecture of the OZIX operating system. Its purpose is to provide a conceptual view of the OZIX system by illustrating the relationship between the various components of the system. The functional components (subsystems and the nub) of the OZIX architecture are described in detail, together with the three basic architectural concepts: the memory object model, the executor model, and the security and integrity model.
- **OZIX Network Implementation Architecture**. This document presents an overview of the network strategy and architecture, followed by detailed descriptions of OZIX's Internet, DECnet, and network management components. Support for Network File System (NFS), remote procedure calls (RPC), IBM interconnections, application programming interfaces (API), and network services are also examined in detail.
- **OZIX System Administration Architecture**. The architecture for OZIX system administration, based on the Enterprise Management Architecture (EMA), is outlined in this document.
- **OZIX User Environment**. This document describes the architecture of the OZIX user environment and a strategy for its development. In the scope of the document, the OZIX user environment is defined by two of its key components: the OZIX run-time libraries, and the general purpose commands and tools available to all OZIX product users.
- **OZIX Security Overview**. This document addresses the security issues involved in the design and development of OZIX products. It discusses the security goals for OZIX products, defines the aspects of OZIX product security, and outlines the approaches used to reach those goals.

June 1989

OZIX Market and Product Requirements Document

Revision number: 1.0

Prepared By:

Terry Morris, *OZIX Product Manager (DECWET::MORRIS)*

Contributors:

Kevin Breunig,*SSGBPM*
Stephanie Parish,*SSGBPM*
Rob Shuster,*OS/SB Documentation*
Debbie Walkowski,*OS/SB Documentation*

Approved by:

Roger Heinen— OZIX Group Manager

Glen Johnson—OS/SB Program Manager

Rick Berzle—Manager OS/SB Product Managers

Digital Equipment Corporation
Confidential and Proprietary

Trademarks of Digital Equipment Corporation

Concert Multithread	DECwindows	VAXcluster
DEC	Rdb/VMS	VAX DOCUMENT
DECnet	Rainbow	VMS
DECprint	ULTRIX	digital™
DECserver	VAX	

Other Trademarks

386 is a trademark of Intel Corporation
Ada is a trademark of the Department of Defense
AIX is a trademark of International Business Machines Corporation
Apollo is a registered trademark of Apollo Computer, Inc.
Apple is a trademark of Apple Computer, Inc.
IBM is a registered trademark of International Business Machines Corporation
INGRES is a trademark of Relational Technology Inc.
Macintosh is a registered trademark of Apple Computer, Inc
Microsoft is a registered trademark of Microsoft Corporation
MIPS is a trademark of MIPS Computer Systems Inc.
Motif is a trademark of Open Software Foundation, Inc.
MS-DOS is a registered trademark of Microsoft Corporation
MVS is a trademark of International Business Machines Corporation
Network Computing Kernel is a trademark of Apollo Computer, Inc.
Network Computing Software is a trademark of Apollo Computer, Inc.
NFS is a trademark of Sun Microsystems Inc.
Open Software Foundation is a trademark of Open Software Foundation, Inc.
OSF is a trademark of Open Software Foundation, Inc.
OSF/1 is a trademark of Open Software Foundation, Inc.
OSF/Motif is a trademark of Open Software Foundation, Inc.
POSIX is a trademark of the Institute of Electrical and Electronic Engineers Inc.
PostScript is a registered trademark of Adobe Systems Inc.
SUN is a trademark of Sun Microsystems Inc.
UNISYS is a trademark of UNISYS Corporation
UNIX is a registered trademark of American Telephone and Telegraph Corporation
X11 is a trademark of the Massachusetts Institute of Technology
X/Open is a trademark of the X/Open Group
X Window System is a trademark of the Massachusetts Institute of Technology

OZIX SYSTEMS PROJECT TEAM

OZIX Engineering Manager:

John M. Gilbert, *OZIX Engineering Manager*

OZIX Engineering Team:

Benn Schreiber, *OZIX Project Manager*
Marilyn Fries, *OZIX Development Supervisor*
Dick Funk, *OZIX Development Supervisor*
Mark Ozur, *OZIX Development Supervisor*
John Penney, *OZIX Development Supervisor*
Mike Peterson, *OZIX Development Supervisor*
Dave Snow, *OZIX Development Supervisor*
Bill Watson, *OZIX Development Supervisor*
Lu Anne Van de Pas, *OSG Compiler Development Supervisor*
Steve Jenness, *OZIX Network Architect*
Chris Saether, *OZIX ABA Architect*
Jim Schirmer, *OZIX Security Architect*
Claire Cockcroft, *OZIX Internationalization Program Manager*
Pete Benoit, *OZIX Project Leader*
Richard Brown, *OZIX Project Leader*
Sumanta Chatterjee, *OZIX Project Leader*
Jan D'Addamio, *OZIX Project Leader*
Mark Ditto, *OZIX Project Leader*
Dennis Doherty, *OZIX Project Leader*
Min-Chih Lu Earl, *OZIX Project Leader*
Debbie Girdler, *OZIX Project Leader*
Kelly Green, *OZIX Project Leader*
Jeff Havens, *OZIX Project Leader*
Brett Helsel, *OZIX Project Leader*
Oscar Newkerk, *OZIX Project Leader*
Charles Olivier, *OZIX Project Leader*
Jim Teague, *OZIX Project Leader*
Dave Walp, *OZIX Project Leader*
Charlie Wickham, *OZIX Project Leader*

OZIX Finance:

Judy Fox, *OZIX Business Support Manager*
Maggie Schimpf, *OZIX Financial Manager*
Salley Anderson-Teague, *OZIX Financial Analyst*

Product Management:

Mike Parker, *Manager OZIX Product Managers*
Kathy Appellof, *OZIX Transaction Services Product Manager*
Terry Morris, *OZIX Product Manager*
Cathie Richardson, *OZIX C & Third Party Applications Product Manager*

Manufacturing:

Rose Ramsey, *Software Manufacturing Product Manager*

Documentation:

Jim Jackson, *OZIX Publications Manager*
Liz Hunt, *OZIX Application Programming Environment Project Leader*
Marcia Aguero, *OZIX System Management Environment Project Leader*
Bill Talcott, *OZIX Software Support Environment Project Leader*
Cheryl Snyder, *OZIX Usability Engineering/Testing Project Leader*

Customer Services:

Bill Hilton, *OZIX Customer Services Systems Engineering Manager*
Thomas Siebold, *OZIX CSSE Maintainability Engineer*
LeeAnn Stivers, *OZIX CSSE Product Manager*
Myrna Harrison, *OZIX ESDP Project Leader*

TABLE OF CONTENTS

FIGURES

TABLES

Preface

The OZIX Market and Product Requirements Document describes the open systems market needs and the specific product attributes required to meet the needs of the market.

Associated Documents

1. OZIX Applications Requirements Document
2. OZIX Version 1.0 Business Plan
3. OZIX Customer Services Requirements Document
4. OZIX Master Documentation Plan
5. OZIX Product Model Document
6. OZIX Serviceability Requirements Document
7. OZIX Usability Plan
8. OZIX Vision Document

Change History

Date	Issue #	Description
February 1989	0.1	Preliminary OZIX Market and Product Requirements Document
March 1989	0.2	Preliminary OZIX Market and Product Requirements Document
April 1989	0.5	Incorporate changes from OZIX project team comments
April 1989	0.7	Add section from SSGBPM
May 1989	0.9	Final internal review before wide distribution
June 1989	1.0	First widespread distribution

Date of Printing:

27-OCT-1989 16:39:41.84

1 Introduction

This document examines the opportunity for DIGITAL to establish itself as a leading systems supplier in the production environment, open systems market.

1.1 Executive Summary

The production systems environment is divided into two (2) distinct segments, proprietary and open systems. DIGITAL must have products for both types of customers. Currently, VMS answers the need for proprietary production systems where the primary competition is IBM MVS large systems and AS/400's in the mid-range. However, in the open systems segment DIGITAL currently does not offer what customers perceive to be a commercially viable alternative to competition such as IBM's AIX, AT&T's System V, HP's UX, and Pyramid's OSx.

DIGITAL now has a window of opportunity to establish itself as a major vendor in the production environment, open systems market by offering industry-leading open systems products. This document, as organized, describes the market requirements, the primary competition, and specifically what functionality is required to be an industry-leading product in this market.

1.2 Open Systems Market Summary

In commercial, technical, governmental, and educational institutions today there is a growing demand for open systems based on the UNIX[1] operating system to support production applications. Production applications are usually characterized as "mission-critical" and include applications such as transaction processing (TP), decision support, laboratory monitoring and many others.

The total worldwide market for UNIX-based systems is projected to grow at a compound annual rate of 20% through the year 1992. UNIX systems supporting commercial applications are expected to reach $12.8 billion in revenue worldwide by 1992 (Dataquest, November 1988, Initial System Sales).

The lack of commercial features in UNIX has already changed and will contintue to change dramatically in the next few years. UNIX vendors that already provide, or have stated intent to provide, a commercial UNIX operating system product are Pyramid, IBM, Tandem, AT&T, Sun, Hewlett-Packard, UNISYS, the OSF, and NCR.

Customers are attracted to open systems because of the following:

- They can build portable applications to a standard set of application interfaces (investment protection)
- They can avoid costly conversions and retraining when needs expand beyond their original system (scalability)
- They can become less dependent on specific component vendors (source independent)
- They can capitalize on innovative, cost-performance platforms

Production environment functionality that customers expect from an open operating system include:

- Production environment quality and reliability
- Department of Defense rated consistent and configurable distributed security model
- Compliance with standards, such as X/Open branding, the Open Software Foundation (OSF), POSIX, and so forth
- Superior system and application availability, fault tolerance, and data integrity
- Providing a high-performance platform for TP and database products

[1] UNIX is a registered trademark of AT&T

- Incorporation of elements from third-party sources
- Integrated system and network administration for low cost of ownership and ease of management
- High performance system and I/O components
- Availability worldwide at FRS for multinational corporation deployment
- Superior documentation
- Centralized software distribution with distributed licensing capabilities
- Superior interoperability and communications

2 Market Definition and Requirements for an Open Systems Production Environment

Production systems are commonly found in business computing. This is the style traditionally dominated by IBM and other mainframe vendors. The production systems market encompasses several traditional styles, including multiuser systems, TP, fault tolerant computing, and software products for security and system management.

Production systems must be highly available and reliable. In many cases, customers use production systems in the daily operations of their businesses. Most production system applications use very large databases. Thousands of users may be deployed on a variety of systems often requiring 24 hours/day, 7 days/week, 365 days/year continuous operation. The systems, users, and databases may be distributed worldwide.

Production systems are characterized by several common attributes, listed as follows:

- Applications that are critical to the profitability or service level of the customer
- Typically large, mature applications involving large databases and supporting a large number of users
- Specific requirements regarding availability, throughput, systems management, security, etc. In an era of global markets, 24-hour system availability is in increasing demand
- Technically unsophisticated end-users

Virtually all systems require some production environment attributes. There are, however, many attributes that distinguish production systems from non-production systems. These are outlined in Table 1.

Table 1: Production Systems vs. Non-Production Systems

Production Systems	Non-Production Systems
Availability critical (99%+)– downtime has immediate impact on profits or results	Availability desired, but limited downtime (< 1 hr.) tolerable
Mature applications running continuously or on regularly scheduled basis	Development environment or mature applications run on occasional basis
Complex job scheduling requirements	Ad-hoc scheduling
Limited batch scheduling window; restart capabilities critical	Batch jobs can be easily postponed, restarted
System monitored and tuned during the day to balance workload, ensure consistent response time	System tuned at installation
Centralized system management by dedicated operations staff	System management handled mostly by end user
Large number of devices (tape, disk, printer) managed by system	Typically fewer devices managed

Examples of production environment applications include the following: accounting systems, payroll, accounts receivable, computer aided software engineering (CASE), computer aided design/manufacturing (CAD/CAM), automated teller machines (ATM) systems, office automation (electronic mail, basic word processing), lab monitoring, some academic applications, communications switching, high-end publishing, modeling, defense command and control systems, decision support, and traditional transaction processing applications such as reservation systems, banking, trader workstations.

Note that a distinction is not made between commercial and technical applications. While most production systems today are used for commercial applications, there are many technical applications that require high levels of availability and throughput (CIM, patient monitoring systems, environmental control systems).

2.1 Open Systems

Customers have a large investment in solutions based on out-dated technology. Even though these systems are expensive to maintain (application maintenance, system management, power consumption, floor space, air conditioning), customers are hesitant to convert these applications to newer technology. They are faced with:

- The cost of re-engineering without substantially increasing functionality, no added-value
- The cost of retraining the user community, both application and cultural
- The cost of running a parallel operation while they transition the systems

They believe open systems allows them to convert only once, thus justifying the investment. Customers believe that open systems provide the vehicle for the easy incorporation of new technologies. Digital can exploit this opportunity by providing a complete open systems product.

An open systems environment is a comprehensive, consistent set of international technology standards and profiles, specifying interfaces and supporting formats for interoperability or portability of applications, data and people. The field of open systems is a continuum of systems, ranging from closed systems that implement no standards to open systems that implement a rich set of standards.

Market demand for open systems is heavily influenced by many open software organizations such as X/Open, OSF, POSIX, OSI, and so forth. One of these organizations, the OSF, was founded in May 1988 to develop, license, and distribute an open software environment that is based on an independent UNIX-derived operating system and related subsystems. OSF addresses three major issues in the open systems market: portability, interoperability, and scalability. Applications written to the OSF standards are portable from one architecture to another. Interoperability provides the ability to exchange data and run applications on computers from multiple vendors. Scalability provides the means to migrate software across different hardware classes. By specifying an environment built on standards that provides these capabilities, the OSF fulfills customers' demands to protect their long-term software investment.

DIGITAL's commitment to the OSF is significant in scope and heavily impacts our long-term strategy in the open systems market. DIGITAL must show a commitment to the OSF by providing systems compliant with the open systems standards as set forth by the OSF's Application Environment Specification (AES). DIGITAL must also originate innovative technologies that may be used by standards organizations or the OSF creating new standards where none currently exist.

2.2 Customer Requirements for an Open Systems Production Environment

Customers would like to have the following qualities in an open systems production environment:

- Systems that interoperate in an environment with the following characteristics:
  - — Multivendor installations
  - — Multiple operating system installations
  - — Multivendor networks
- Coordinated multivendor access to data with atomic transaction control
- Distributed processing
- Scalable high-performance

- High availability, high reliability, and fault tolerance
- Fast, efficient, and high-quality service
- Well-defined Standards
- Tools, languages, and portable applications (from DIGITAL and third party suppliers) available at first revenue shipment (FRS)
- Enterprise-wide system and network administration
- High quality, easy-to-use documentation
- Secure systems
- Fully internationalized systems

In addition, the vendor of choice must provide a business environment that is much like a partnership. The vendor must be easy to deal and do business with by establishing trust and long term relationships with the customer. The vendor must also provide a suite of fully integrated applications, vendor and third party developed, which solve the customers business problems.

2.2.1 Multivendor Environment

The operating environment of an open systems customer includes a variety of systems and networks from many different vendors. The variety of systems range from PC's to mainframes and everything in between. The customer must also deal with a spectrum of proprietary and industry standard operating systems, networks, data formats, and so forth.

The problem is in linking different systems, networks, and applications together to provide integrated management of, and to facilitate the transparent flow and exchange of data (records, files, messages, and so forth) between these different systems, networks, and applications.

The problem is in providing integrated management capabilities of all these different systems, networks, and applications and to facilitate the transparent flow and exchange of data (records, files, messages, and so forth).

Customers are looking to the vendors and standards organizations for help in providing answers to this dilemma.

2.2.2 Distributed Processing

The trend is to place the compute power (systems) where the compute cycles are needed. By definition, the distribution of compute power requires remote deployment of data, hardware, operating system software, and applications software. Even though these systems are distributed (often provided by different vendors) they are expected to be managed or controlled and function as if they were centralized. They must communicate well with each other.

The distributed computing philosophy puts an added burden upon the customer's human and network resources. The burden on the human resources is in providing system and network management by non-technical personnel. The burden on the network is in providing the bandwidth to effectively use all of the available compute power. In other words, to use idle compute cycles regardless of the physical location. Customers want easy and efficient access to all of their available compute resources, wherever the resources are located.

Configurations vary widely within installations. Development and scientific or technical environments tend to use back-end file and compute servers for workstation-based development. Transaction-based environments use a more conservative centralized approach with character cell terminals or PCs connected to a large minicomputer or mainframe holding the corporate databases.

Although the trend in both the TP and development environments is toward a workstation to server relationship, customers do not see the need for character cell terminals support to diminish until well into the 1990s. Character cell terminals are pervasive throughout the industry and, for many customers, replacing them with workstations is not economically feasible.

Workstations are usually purchased based upon a specific applications availability. Workstation vendors include: Apollo, Sun, Silicon Graphics, DIGITAL, HP, IBM (PCs and compatibles), and so forth. Servers are, therefore, expected to support workstations from a variety of vendors. The server can provide a batch compute facility, a print facility, a file or database repository, or workstation installation and/or boot support.

2.2.3 High Performance

Customers are constantly needing more performance from their computer systems. They believe that price/performance will double every two years. Hardware performance from a uniprocessor or multiprocessor system, however, is not the only measurement. With hardware quickly becoming a commodity, software performance is emerging as the true measurement that differentiates systems. Customers want software features that improve performance for applications as well as the operating system.

Performance is measured in many ways, one of which is by the system I/O bandwidth and throughput. This is both a function of the hardware and the cooperating I/O software. The I/O subsystem software includes network, terminal, and disk and tape operations. Systems supporting production applications, especially high transaction rates, need high bandwidth hardware and fast, efficient, and flexible supporting software together providing maximum throughput.

Another means of improving performance may be realized through the use of parallelism and concurrency in applications themselves. This type of improvement often requires hardware support for multiprocessors and software support in compilers (decomposition) and the operating system (threads). However, from a price/performance perspective adding a processor and memory is much more cost effective than complete hardware replacement.

Performance measurement software for predicting real and potential bottlenecks in the operating system, related subsystems, and application software is absolutely necessary for this environment. These tools are abundant for proprietary systems currently sold into the production system environments. Tools that support and can provide performance data on decomposed applications and multiple thread environments are not widely available as yet.

2.2.4 Scalable Systems

Needs vary widely between application and system location deployment. Systems need to be scalable from small systems providing limited functionality to large systems providing a broader spectrum of services without sacrificing performance or functionality. The need is really a function of growth potential. Customers need to be provided a means to grow from small to large without swapping hardware equipment and software and without having to go through costly conversions.

Scalable systems provide the customer with investment protection for their most costly resources, application software and personnel. This translates directly to low cost of ownership for the systems.

Scalability may be provided in the following ways:

- Supporting more processors, memory, disk, and so forth on a single system
- Coupling multiple systems together as a single system
- Coupling multiple systems together in a distributed, wide area network
- A combination of any or all of these methods

2.2.5 High Availability, High Reliability, and Fault Tolerance

Customers expect hardware and software failures to be automatically detected with speed and accuracy. The failing component is expected to be isolated, taken out of service, and replaced with little or no impact on the users, applications, or system. While a failed component is replaced, customers expect a predictable, graceful degradation of their system rather than total interruption of service to their users. The maximum acceptable number of hardware or software service interruptions in this market is one or less per year.

Software failures or crashes are not expected to interrupt service. In the rare case when a hard crash does occur, the operating system should provide a graceful shutdown, 100% file system integrity, and a fast reboot/restart. At the high end of the market, customers expect the vendor to provide a failover mechanism to prevent total service interruption.

Application restart at the point of failure is usually provided by the applications themselves, especially in *mission-critical* situations. This is a timeconsuming and sometimes difficult task. Operating system supported (embedded) mechanisms providing this type of capability are gaining in popularity as the number of mission-critical applications grows. This is a feature found in few open systems today.

A computing environment that provides for uninterrupted operation is rapidly becoming a requirement in some of the upper mid-range configurations today. By the mid-1990's it is expected that vendors competing in the upper mid-range to lower high-end production systems will be required to provide this feature.

Commercial file systems are expected to be 100% safe from corruption. Due to the sensitivity of the data the file system is expected to provide flexible, quick, and reliable access to the data. Customers desire a powerful and flexible file system with high security, data integrity, and recovery capabilities. Access methods vary from application to application with expectations that the system provide *standard* access methods such as flat, ISAM, and database management techniques. In addition, many customers feel that access to other types of file system interfaces, such as Apple, MS-DOS, System V RFS, and DIGITAL's ODS-II are very important as well.

2.2.6 Fast and Efficient Service

It is assumed that customers who purchase systems in the next 5-10 years for use in upper mid-range and high-end of the production environments expect the vendor to supply a zero downtime, fault tolerant, 100% reliable computing system. Systems acquired for use in low-end to mid-range production environments are somewhat less restrictive but still very strict compared to most other markets. When a problem with the hardware or software does arise, production-system customers expect a level of service greater than that currently supplied by vendors to the traditional UNIX customer.

Initial installation of the software should be a trivial exercise, that is, it should be fast and efficient with little or no operator intervention required. Subsequent updates and upgrades of both operating system software, tools (layered products), and applications are expected to be accomplished with little or no interruption of service to the end-user.

2.2.7 Well-defined Standards

Customers want standards that are well-defined, published, and accessible. Standards allow changes to be made in an orderly manner rather than at the vendor's discretion. They are considered essential to end users and third-party software vendors who build platform-independent applications. Governments, both foreign and domestic, require that certain standards be met before the vendor can respond to issued requests for proposals (RFPs).

2.2.8 Tools, Languages, and Portable Applications

Tools, languages, and applications that help provide solutions for customers business problems must be available at first shipment. The solutions (software) must be state-of-the-art and fully support features of the hardware and operating system software. Development and performance monitoring tools must be available for tuning not only applications but also the operating system and networks. Customers require more production-oriented tools to improve productivity and efficiency. There is a great demand for tools that cover the following areas: CASE, code debug, profiling, application prototyping, optimization, configuration management, databases, 4GLs, integrated system and network management, languages (C, C++, COBOL, FORTRAN, Ada), editors, graphics and math libraries, and communications. In addition, many customers buy their hardware based on a particular application package needed to give them a competitive edge.

2.2.8.1 Third Party Application Software

Application software sells systems. The initial system sale is contingent upon the spectrum of available software for a given system. Quite often the software of interest to the customer already exists within the customers environment. As a stipulation of sale, the software must also be available on the new machine as well. A potential computer purchaser is interested in the breadth and quality of the software that is available for any given machine. This is often a measure of acceptance by the software development community as well as a sign of maturity of the product.

2.2.9 Enterprise-wide Administration

The system and network administration of a distributed computing environment is difficult to use and usually requires a large staff devoted exclusively to these tasks. Administration is a sizable investment for customers. To lower the overall costs associated with system and network administration a universal, tailorable, and expandable tool, for both multivendor network and system administration, is needed. Customers would like a consistent single easy-to-use interface allowing them to manage systems and networks from a single location. This tool would let the system or network administrator manage servers (compute, file, database, TP, etc.), workstations (disk and diskless), and a variety of networks (LAN, WAN, OSI, SNA, TCP, X.25, and so forth) with a small staff, reducing training costs. Customers refer to this kind of administration as "lights out", "unattended operation", or "auto-pilot".

Installation, updates, and upgrades of the system and layered application products are important aspects of system administration and should occur with as little disruption as possible to the operation of the system. UNIX systems have traditionally been known for their difficult installations. Sun Microsystems has clearly set a standard for UNIX systems with their 386i installation program, The Organizer. This product, aimed directly at the non-technical user, demonstrates the type of usable installation commercial customers would like to see for larger systems.

Customers and third party software vendors are also very interested in centralized software distribution supporting a distributed software licensing mechanism. Benefits to the customer include the following:

- All software for a particular system would be distributed to the customer on one distribution media
- All software would be administered from one site eliminating tape and disk storage, multiple installations of the same product on many machines greatly reducing administration costs
- Software updates and upgrades are simultaneous on all systems
- Documentation can be distributed in a similar fashion

System performance management is very important as well, especially in TP environments where system loads can change dramatically in short periods of time. Systems in TP environments are expected to have consistent predictable performance that can be accurately measured. Tools that accurately and consistently predict peak load performance are needed. Customers in transaction environments often purchase their equipment based on such performance.

Customers also want a mechanism for accurately dispensing system resources to particular jobs or users. Often, users or departments "fund" a percentage or portion of the system. The system administrator must be able to allocate the system resources proportionately to the appropriate users or applications.

2.2.10 Commercial Environment Features: Print, Batch, and Media Management

Efficient print and batch management is extremely important in production environments. Typical production environments have high volume batch requirements. In many such installations batch processing may account for the vast majority of computer usage. The batch processor must take advantage of lightly loaded machines in the network without manual intervention from an operator or user. Operators or users must have the ability to examine and modify the status of a job regardless of the location or execution of the job. Printing needs vary from low-speed, low-quality printing of reports on a PC (dot-matrix) to very high-speed (100+ pages/minute) jobs that require sophisticated laser printers. Production environments often have tightly controlled printing functions, such as the printing of payroll or accounts payable checks. Printing servers must be able to accomplish the following:

- Channel specific jobs to specialized printers (for example: checks to an impact printer)
- Schedule jobs
- Examine and modify the printing parameters
- Restart the job on a particular line or page, or on a different printer
- Divide a print job between multiple printers
- Print documents that contain both text and graphics
- Balance the printing load
- Provide accounting information

Commercial installations typically have large media management problems that have often been automated either from acquisition of third party or internally developed software. A subset of the functions of the software include the following:

- Migration of frequently used data to fast storage
- Migration of less frequently used data to slower storage
- Automatic archiving of infrequently used data
- Automatic retrieval of archived data
- Library management tools

2.2.11 Documentation

Commercial customers have categorically stated that the current level of documentation is confusing, difficult to use, and does not meet their needs. It is not usable and has not improved as systems have become more sophisticated. Most end users are novice computer users and require explicit, easy-to-find, concise information with many examples.

Many customers have expressed a great deal of interest in lowering their documentation costs. Customers want the following:

- Lower overall documentation costs (reduced number of hardcopy documentation sets)
- Updates in a more timely fashion and do not want to "replace pages" in current manuals
- Reduced training costs
- Ability to *customize* the documentation for specific needs

2.2.12 Secure Systems

With the proliferation of worldwide computer networks and the increasing number of security violations, customers have realized that they need more secure systems. By 1992 the United States government will require a C2 rating for all computer systems purchased. Many commercial customers are already leading the government in security requirements. This trend toward increased security is expected to continue with both government and commercial customers; by the mid-1990s, users will require a B2 (or equivalent) rating. However, many customers also want the ability to configure the security parameters to meet their various needs.

2.2.13 International Systems

Customers who purchase large TP systems are primarily large multinational corporations, many with worldwide operations. Systems developed by these corporations are usually not just local to one district, region, geographic area, or country. The systems are developed to be deployed throughout the corporation as soon as possible.

Typically, country-specific porting of the operating system, utilities, applications, tools, and so forth, requires an additional six months to a year of engineering effort after the initial release of the product. Supplying language-neutral support for immediate worldwide shipment at product introduction will give a vendor a distinct competitive advantage. Customers may develop applications that may be deployed worldwide at the customers discretion rather than the vendors.

A second consideration is the growth of the worldwide computer systems market. It is increasingly important that customers from many different countries and cultures can use new systems without translation problems. Currently, greater than half of DIGITAL's income is derived from sales outside the United States. At present, the three largest computer markets (in order of ranking) are the United States, Japan, and France. DIGITAL's market share in Japan is approximately 1.5 percent. Without Asian and European language support DIGITAL will not be able to increase marketshare in Japan.

3 Open Systems Market Analysis

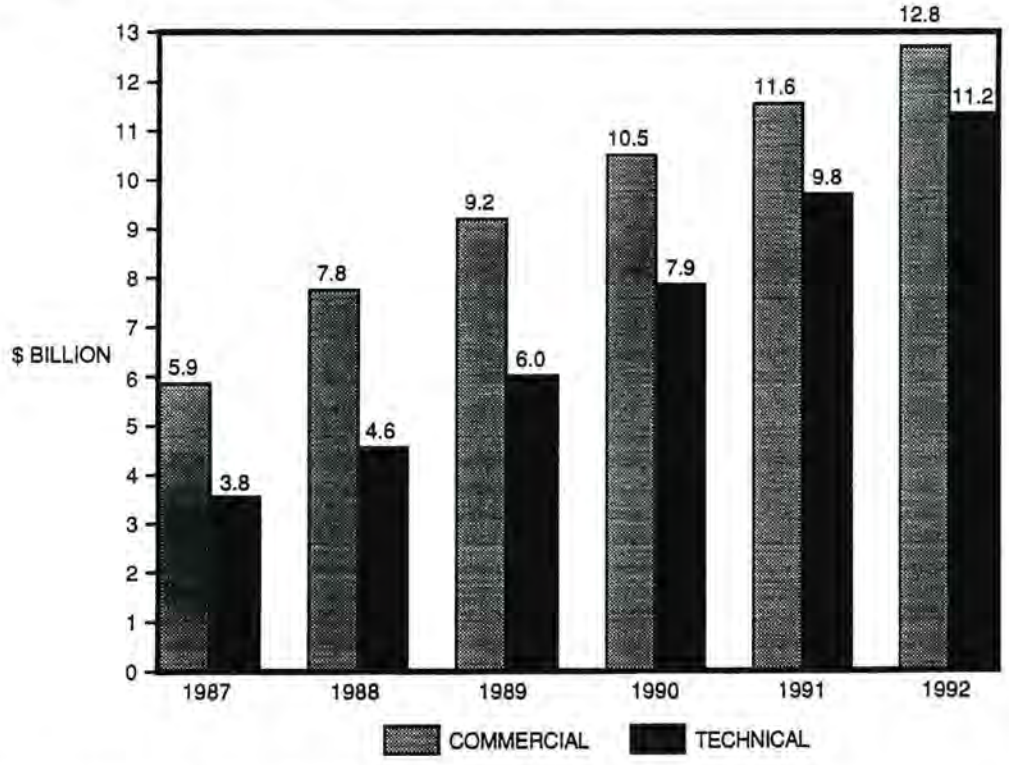
The market for UNIX-based systems represents one of the best opportunities for selling production systems. Within the UNIX market, companies depending on commercial applications will be most attracted to the functionality offered by production systems. UNIX commercial application customers choose open systems for several reasons:

- Hardware vendor independence
- Applications portability
- Data transparency
- Strong distributed environment
- Investment protection in hardware, software and training
- Typically better price/performance than proprietary systems

Additional production application requirements include high security, high availability, high throughput, and ease of management. A growing number of technical applications also require these features.

The commercial UNIX sector is a substantial market. UNIX systems running commercial applications exceed those running technical applications on a worldwide basis with revenues reaching $12.8 billion in 1992. The compound annual growth rate of commercial UNIX systems sales is projected to be 16.8% between 1987 and 1992, which is more than twice the growth rate of the overall computer systems market.

Figure 1: Worldwide UNIX System Sales by Application Segment $B (if-sold value)

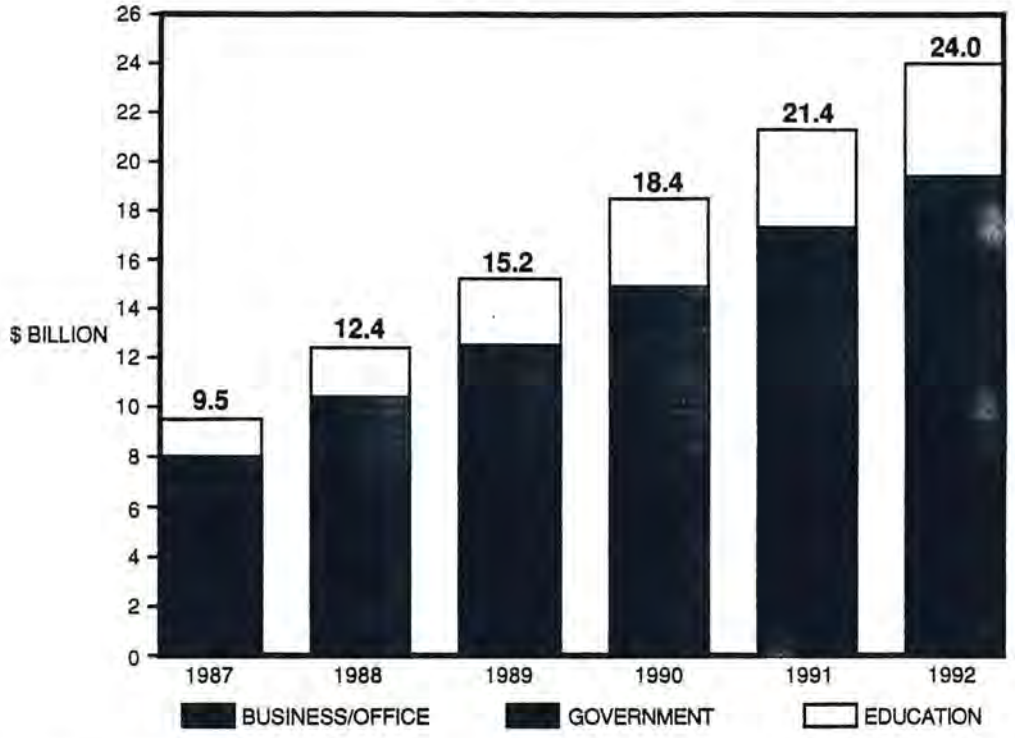


Source: Dataquest, November 1988

The above forecast was made prior to the availability of a production quality UNIX system, a factor which would accelerate UNIX demand.

Government sectors (both domestic and foreign) represents another opportunity for UNIX production systems. Many requests for proposals issued by the government now require systems to support the definition of open systems and standard interfaces. An estimated 70% of government RFPs now specify UNIX in some form. The government sector using UNIX systems represents approximately one-third of the UNIX-based systems sold today and will maintain that share of the market into the 1990s.

Figure 2: Worldwide UNIX System Sales by User Environment $B (If-sold value)



Source: Dataquest, November 1988

3.1 UNIX-based Applications

The potential size of the UNIX commercial market is highly dependent upon the availability and quality of applications that meet the requirements of commercial computing. Availability of applications, however, is in turn dependent upon the availability of high-quality commercial CASE tools.

Although market information on this topic is scant, International Data Corporation (IDC) performed surveys to identify the key applications running in both commercial and technical markets. The results of the commercial application survey are summarized below. The System Software Marketing (SSM) group is continuing this analysis with an end-user survey of open systems applications, to be completed by the end of FY89. SSM's analysis will be included in future revisions of this document.

Table 2: UNIX Applications at U.S. Commercial Sites, 1987 (Sample = 165 sites)

Primary Application	Number of Responses	Per-Centage	Leading Vendor	DIGITAL's Position
Text Processing	189	13.2%	AT&T	3
DBMS	185	12.9%	AT&T	3
Accounting	150	10.4%	Altos	NA
Spreadsheet	148	10.3%	AT&T	NA
Communications	123	8.6%	Altos	4
Software Development	108	7.5%	Altos	3
Integrated Office System	80	5.5%	Altos	NA
Payroll/Personnel	80	5.5%		
4GL's	77	5.4%	AT&T	NA
Sales/Distribution	64	4.5%		
Graphics	49	3.4%		
Other	184	12.8%		
Total	1437	100.0%		

Source: IDC, March 1988 (some sites include multiple responses)

Although production systems have the greatest opportunity in commercial applications, there are a growing number of "technical" applications requiring production systems features, such as high availability, archiving, and systems administration. Examples of these application areas are communications switching, lab monitoring systems, and process control.

3.2 Production Systems Market

The production systems market spans several traditional market boundaries, all of which are shifting.

Table 3: Production Systems Market Sizing

Segment	1988 ($M)	1992 ($M)	CAGR ('88-'92)	Source
All Multiuser Systems	58,848	73,966	5.9%	Dataquest 7/88
All OLTP Systems	31,415	50,858	12.7%	Infocorp 4/88
All Fault Tolerant Systems	3,113	6,907	22.0%	Frost & Sullivan 12/88
All UNIX General Purpose Systems	9,549	16,633	14.9%	Dataquest 7/88
Data Center Automation Software	2,562	5,800	22.0%	Input 12/87

Note: Segment data given shows relative size and growth of each segment. Because of differences in definitions across sources, segment numbers are not directly comparable. SSM is in the process of sizing the production environment open systems market based on applications market data and assumptions about its penetration.

An analysis of mainframe class computers (value of $1 million and above) running commercial applications indicates that the vast majority require production attributes.

LEADING APPLICATIONS ON BUSINESS MAINFRAMES

RANK	APPLICATION
1	Accounting*
2	Payroll/Personnel*
3	Order Processing/Inventory Control*
4&5 (tie)	Sales/Distribution*
	Purchasing*
6	Other
7	Manufacturing*
8	Mathematics & Statistics
9	Engineering
10	Banking*
11&12 (tie)	Insurance*
	Education

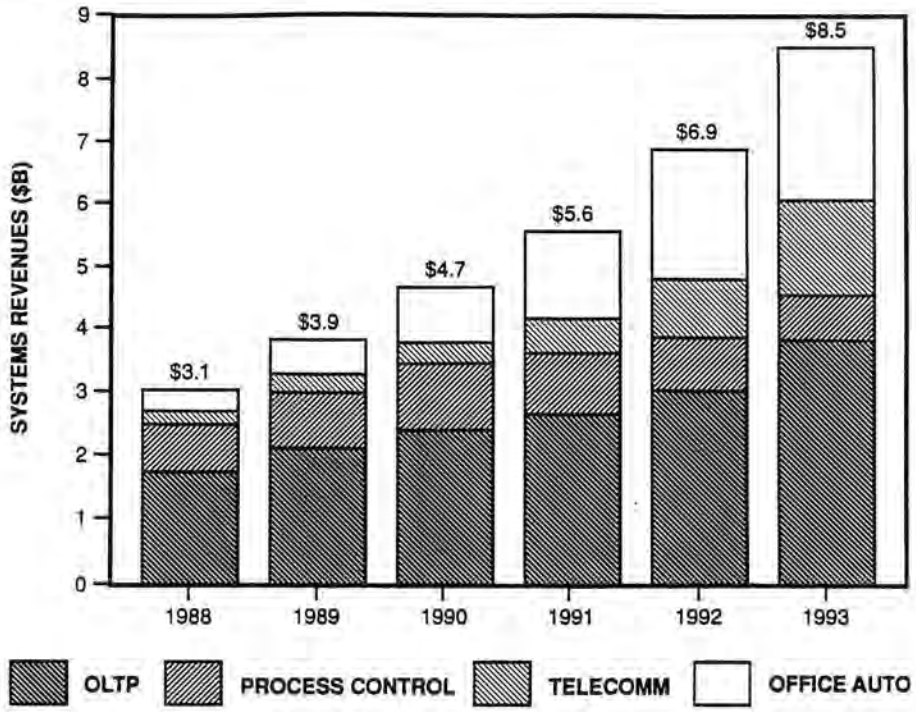
(* typically production applications)

Source: Dataquest 12/88

TP is growing as a percentage of all production systems. The Yankee Group estimates that by 1991, 85% of mainframes, 65% of minicomputers, and 5% of microcomputers shipped will be used for OLTP. Much of the growth in OLTP is coming from improvements in hardware price/performance, making OLTP affordable for new applications such as manufacturing process control and retail point-of-sale systems.

Fault tolerant systems is the fastest growing segment of the production systems market. As hardware becomes more reliable, competition is shifting toward software and network error detection and recovery. While fault tolerance is often thought of as a TP feature, the fastest growing markets for fault tolerance are telecommunications and office automation. DIGITAL can clearly differentiate itself by providing a low cost, fault tolerant file server for office environments.

Figure 3: Fault Tolerant Systems Market 1988-1993



Source: Frost and Sullivan

Fault tolerance is also critical to penetrating the largest production systems industry markets. By 1993, the largest markets for fault tolerant systems will be banking (33% of sales), other financial (16%), manufacturing (13%), and communications (11%).

UNIX penetration of the production systems market has been limited. According to Infocorp, the UNIX share of OLTP units shipped in 1987 was 18% for systems in the $50K-and-under price band, and 10% for the $50-350K price band. UNIX penetration in this market to date is due to underlying hardware price/performance or mandated UNIX purchases, not UNIX capabilities in production systems. For UNIX to appeal to a broader production system market, it must overcome several technical limitations of UNIX. These limitations are as follows:

- Performance degradation due to fragmentation of large data files
- Performance degradation due to frequent context switching
- Limited security features
- Lack of tuning and diagnostic tools

In addition, UNIX suffers from several perceptual problems in MIS shops, including its image as a technical "hacker's" environment, its general lack of commercial applications, and a lack of system management tools for large data centers.

To date, vendors have employed several strategies to overcome these limitations in production environments. These strategies include:

- Modify the UNIX kernel to support production applications (Pyramid, UNISYS)
- Add a layered TP monitor to UNIX to provide production support (AT&T's Tuxedo/T)

- Enlist support of relational database (RDBMS) vendors who have added data integrity and TP extensions to their products
- Focus on making proprietary environments more "open" through POSIX operating system interfaces

3.3 Production System Market Segmentation by Industry

Open systems will appeal to different production environment industries based on industry requirements, such as the number of transactions per second, the level of system availability, and open systems policies. Figure 3 shows prioritized industry groups DIGITAL should pursue. This information is based on interviews with DIGITAL's Industry Marketing Groups as documented in the Strategy Integration Project from System Software Marketing, April 1988.

Based on our definition of the production environment, open systems market, the top five (5) industries for potential penetration and acceptance are the federal government, transportation, telecommunications, medical and financial services.

Figure 4: Production Environment Open Systems Market Segmentation by Industry

CUSTOMER REQS.	INDUSTRY MARKETING GROUPS												
	FED	TRN	TEL	MED	FIN	AUT	HEA	ED	F&B	ELE	AE	O&G	CHM
High TP	■	■	■	■	■	■	■		■	■	■	■	
High System Availability	■	■	■	■	■		■		■	■	■	■	■
Appl. on Dedicated Proc.	■	■			■		■	■					
Complex Job Scheduling	■	■	■		■	■							
Critical Data Integrity	■	■	■	■	■	■	■	■		■	■	■	■
Central System Mgt.	■	■		■	■		■	■	■				
Large # of Users	■	■	■	■	■	■			■				
Open Systems Policy	■		■	■		■		■		■			
P/P Systems	■	■	■	■		■	■	■					
TOTAL REQS.	**9**	**8**	**7**	**7**	**7**	**6**	**6**	**5**	**4**	**4**	**3**	**3**	**2**

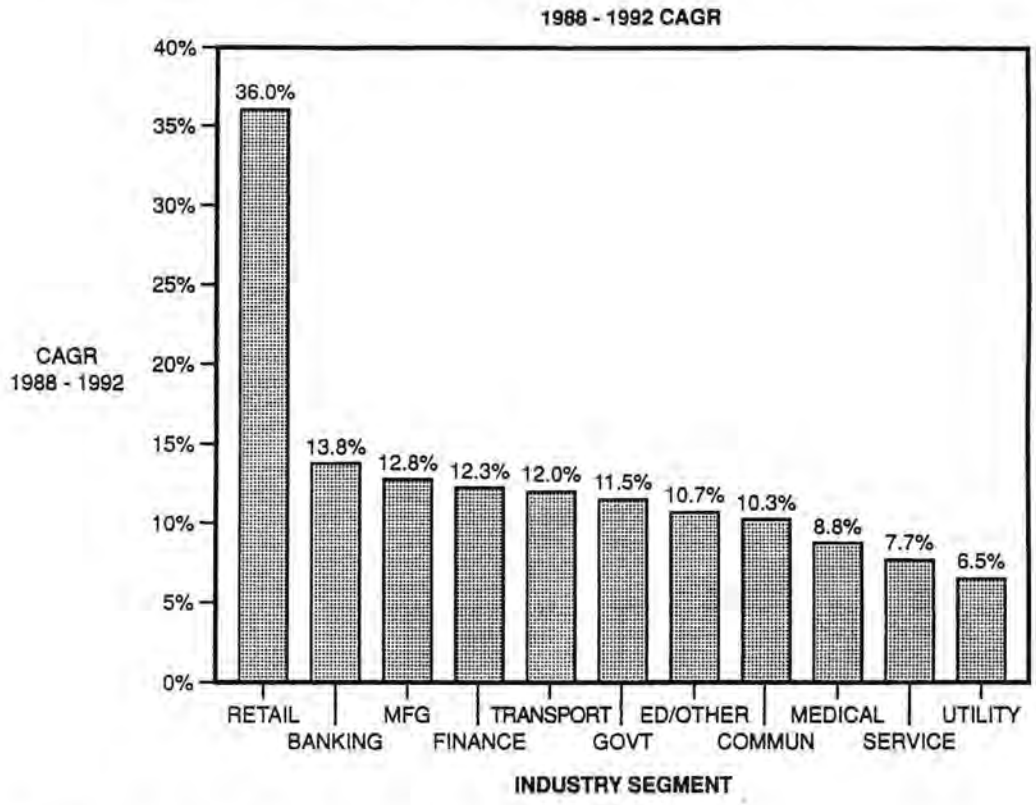
Industry Marketing Group Key

FED = Federal Government
TRN = Transportation
TEL = Telecommunications
MED = Media
FIN = Financial Services
AUT = Automotive
HEA = Health
ED = Education
F&B = Food and Beverage
ELE = Electronics
AE = Aerospace
O&G = Oil and Gas
CHM = Chemical

DIGITAL will initially have to target growth markets where the likelihood of new, standalone applications development is higher. The risk aversion of MIS managers, and existing investment in corporate databases, applications, and personnel training, make it unlikely that they will switch large mission-critical applications to a UNIX platform in the short term. Purchase decisions will be

based upon expectations of a high quality, low cost (to maintain), fully functional system that is very low risk.

Figure 5: Transaction Processing Market Segments by Growth Rate



Source: DIGITAL OLTP Market Definition Taskforce, Sept. 1988

Table 4: Transaction Processing Market Size by Industry Segment

	System Revenues ($B)			CAGR	
	1988	1992	1996	'88-'92	'92-'96
Retail	1.9	6.5	22.5	36.0%	36.4%
Banking	7.4	12.4	20.8	13.8%	13.8%
Manufacturing	6.0	9.7	15.4	12.8%	12.3%
Finance	3.4	5.4	8.8	12.3%	13.0%
Transportation	0.7	1.1	1.6	12.0%	9.8%
Government	1.1	1.7	2.7	11.5%	12.3%
Education/Other	1.6	2.4	3.6	10.7%	10.7%
Communications	2.9	4.3	6.5	10.3%	10.9%
Medical	2.0	2.8	4.1	8.8%	10.0%
Service	2.3	3.1	3.9	7.7%	5.9%
Utilities	1.4	1.8	2.2	6.5%	5.1%

Source: OLTP Market Definition Taskforce

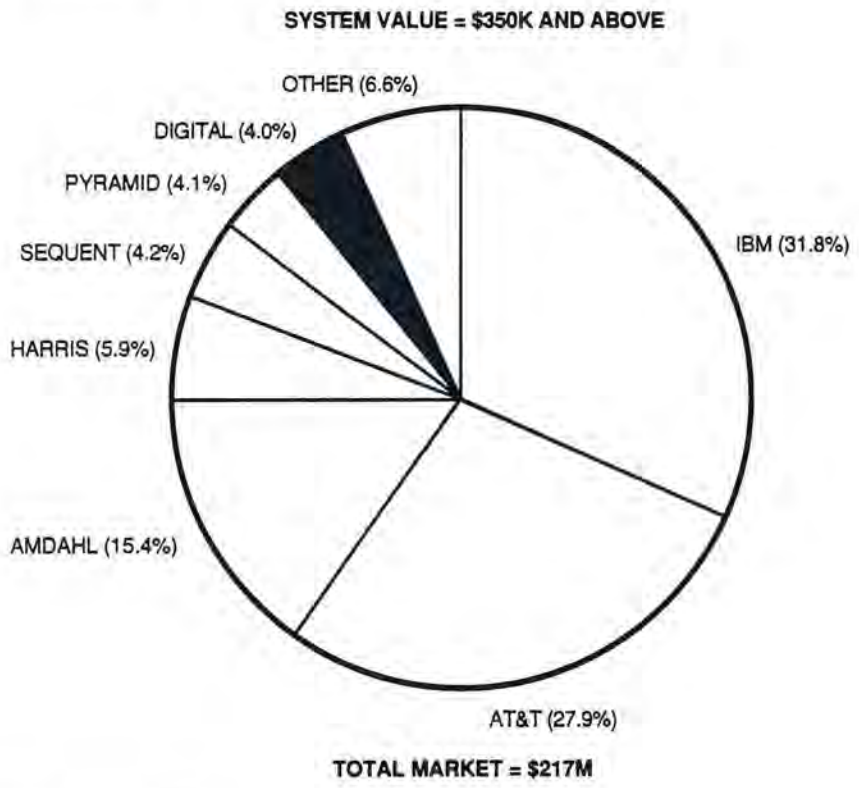
As indicated in Figure 5 and Table 4, the retail, manufacturing, and financial service sectors offer substantial opportunities for selling platforms for new applications. In addition, these industries lend themselves to distributed solutions, where DIGITAL has established strong technical leadership. (The manufacturing growth rate may actually be understated above; Gartner Group estimates the growth rate for OLTP in manufacturing at almost 25%.) Of the three, manufacturing is likely to move most rapidly towards UNIX.

3.4 Competitor Analysis

The window of opportunity for DIGITAL is between 1989 and 1992. A number of vendors will have upgraded their UNIX offerings to support production systems. However, it is important to examine the competitive position of some commercial UNIX vendors today to predict who will present the strongest challenge to OS/SB and DIGITAL in the future.

The competitive landscape for commercial UNIX systems in the high-end is radically different from that for technical UNIX systems. While the market for UNIX systems running commercial business applications is larger overall than the technical market, most of the commercial market growth has been in low-end systems. Infocorp estimates that UNIX penetration of commercial systems valued at $350K and greater was only 2% in 1988.

Figure 6: U.S. Commercial UNIX Market 1988



Source: Infocorp, March 1989

An examination of the U.S. commercial UNIX systems market ($350K and above), a likely DIGITAL target segment, shows a fairly volatile and fragmented market. IBM's market share is misleading, since it is based on revenues from only a half-dozen large systems running IX/370, a third party product. (Shipments of AIX/370 have been delayed until late 1989). AT&T is also in a strong position,

as it is the developer of System V, the preferred UNIX version for commercial and government accounts. While IBM's sales between 1987 and 1988 were fairly flat, AT&T sales to this segment more than tripled from $19M in 1987 to $60M in 1988. This may be due in part to the availability of a CICS-like TP monitor called Tuxedo for AT&T System V. DIGITAL's share was limited to 4% in 1988, in part because only 20% of its UNIX systems were estimated to be used for commercial applications.

More significant, however, is the emergence of Sequent, Pyramid and Sequoia into the number five, six, and nine positions respectfully, after only two years of sales into this segment. All three have grown rapidly by offering strong price/performance, an enhanced UNIX kernel, and fault tolerance to beat general purpose vendors in industries such as finance and telecommunications.

Pyramid is the most direct threat to DIGITAL's entrance into the production environment open systems market, offering a RISC-based multiprocessor server (the MIServer extension of the 9000 series) with a modified UNIX kernel running at a maximum of 140 MIPS and 1000 users. Pyramid dominated the mid-range ($200-350K) commercial UNIX systems market, and held 12.9% of the overall mid-range market in 1988–more than double DIGITAL's share of 6.1%. Infocorp estimates Pyramid controls 50% of the commercial system sales in this priceband.

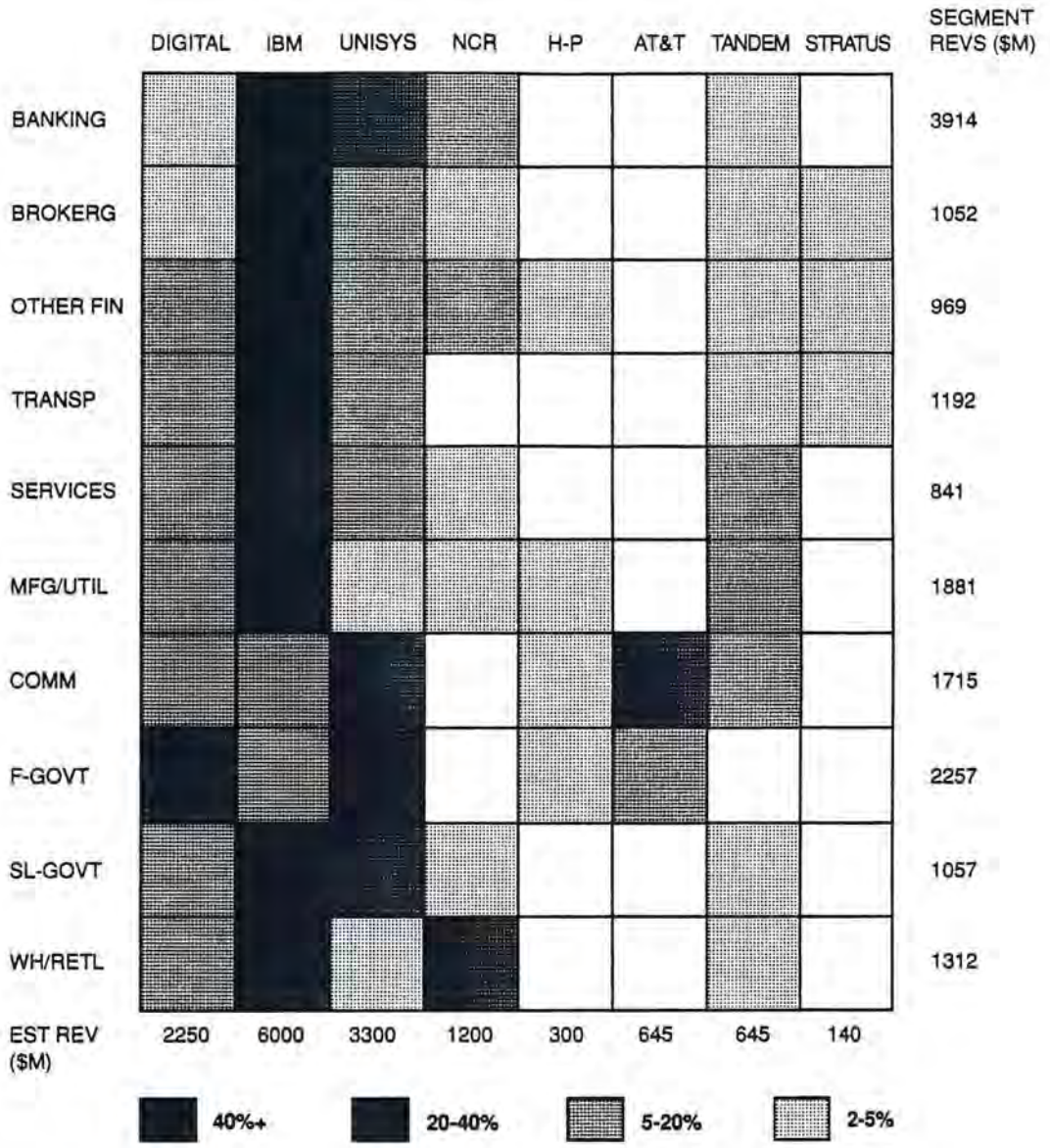
Because Infocorp tracks only manufacturer revenues, UNISYS does not appear in the market share figures. However, UNISYS is a serious threat in this market, and is already advertising itself as the #1 supplier of commercial UNIX systems. It is working with AT&T to enhance System V for commercial environments and is pushing for UNIX standards for TP. UNISYS claims its total UNIX business in 1988 was about $800 million.

IBM, DIGITAL, UNISYS, and Tandem have targeted most segments. IBM dominates most segments with its proprietary platforms. NCR has been successful in targeting two vertical markets: retail and financial services. AT&T has developed a strong presence (26% share) in the communications market.

Arix, Sequent, and Pyramid all have less than one percent share in any one TP vertical market in 1988, International Resource Development, Inc. (IRD). The top vertical market for each are as follows:

- Arix
  - — Communications
  - — Transportation
- Sequent
  - — Banking
  - — Communications
  - — Service
- Pyramid
  - — Banking
  - — Communications
  - — Service

Figure 7: Competitor U.S. OLTP Share by Industry - 1988



3.4.1 Competitor Groups

Four groups of competitors will compete with DIGITAL in the open production systems market, each with a different strategy. The four competitive groups are as follows:

Hot Box Vendors (Pyramid, Sequent, Tolerant, Sequoia, Convex): These vendors, early entrants into this market, are highly focused on providing high performance UNIX platforms with some fault tolerant features based on standard or RISC multiprocessing technologies. These vendors are likely to compete on price/performance in certain applications where availability and UNIX are critical (for example, government and telecommunications).

Fault Tolerant Vendors (Tandem, Stratus): Both Tandem and Stratus, recognizing that the fault-tolerant segment is really a speciality segment (5-10% of TP), are trying to broaden their appeal as general purpose TP providers. UNIX TP is an obvious growth path. Tandem now offers UNIX on a non-fault tolerant platform (LXN) targeted at smaller businesses. Stratus is developing (with Olivetti) a native fault-tolerant UNIX based on System V R3, with planned availability in late 1989. It also offers a UNIX command set (UFS) running under its proprietary VOS operating system. These vendors are likely to leverage their experience selling high-availability, high-performance proprietary systems into the UNIX production systems market.

General Purpose Vendors (DIGITAL, IBM, UNISYS, NCR, H-P, AT&T, Convex): Each vendor is likely to introduce a RISC/UNIX platform capable of handling production systems throughput over the next five years. These vendors are likely to be somewhat conservative, waiting for standards to emerge for UNIX TP. AT&T and UNISYS are in a strong position to bring a production systems UNIX quickly to market by virtue of their cooperation on System V development. While most vendors are likely to accelerate UNIX development for this market, some (like DIGITAL) may hedge their bets by making their proprietary environments POSIX-compliant. Such an environment would appeal to MIS/DP managers who want open standards but also want a stable, mature production systems architecture with a large selection of packaged software. Honeywell Bull is an example of a vendor which seems to be focusing its efforts on POSIX rather than UNIX.

RDBMS Vendors (Oracle, Sybase, INGRES, Informix): DBMS vendors are not hardware vendors but are positioning their products today as a means of getting around the shortcomings of UNIX in production environments. Embedded features in the DBMS handle disk allocation, data integrity, journaling and task management independently of the operating system. Sybase has an embedded transaction server which appears to UNIX as a single process, reducing context switching. Oracle improves throughput by managing data blocks directly, reducing fragmentation. Many RDBMS products run on multiple platforms, so these vendors are likely to position themselves as points of integration between proprietary and open systems. They also stand to become de facto standards in this market as users develop RDBMS applications.

3.4.2 Substitutes

While a large number of commercial users will mandate UNIX for their production systems, others are more interested in vendor independence and price/performance than in UNIX. DIGITAL will compete against several potential substitutes. They are as follows:

- POSIX-compliant proprietary operating systems
- Mach (a UNIX derivative being rewritten by Carnegie-Mellon to support TP)
- Pick

3.4.3 Competitor Profiles (*Preliminary*)

AT&T:

- Offers proprietary TP monitor (Tuxedo/T) running on System V to handle production application tuning, security, and task management encouraging DBMS vendors to support Tuxedo as TP front end.
- Release 4 of System V to support:
  - — Improved system administration.
  - — Convergence of BSD, System V, and Xenix.
  - — Two-phase commit.
  - — Internationalization.

  - Real-time extensions.
  - C1 security rating (B2 expected in release 4.1).
- Release 4 will not support:
  - Disk shadowing.
  - Multiprocessing.
- UNIX International founder.

BIIN:

- Joint venture between Siemens and Intel.
- Offers *selectable* fault tolerance, where processors can be switched from standby to normal mode.
- TP services embedded in the hardware, reducing overhead.
- Limited throughput (12-18 TPS).
- Likely to appeal to government accounts for the following reasons:
  - Operating system written in Ada.
  - "Biin reconstituted UNIX into a robust operating system with the security and integrity of a proprietary operating system..." (Gartner Group, November 1988).
- Primary initial target markets (Gartner):
  - Federal government.
  - Manufacturing.
  - Healthcare.
  - Other mission-critical applications.

CONVEX:

- Traditional superminicomputer manufacturer whose customers say: "Why do I need fault tolerance? I have a Convex and it never breaks."
- Offers high performance storage subsystem (IBIS disks) with disk striping and shadowing.
- Supports large memory configurations (up to 2GB).
- Interoperates well in DIGITAL environment with *Convue Shell* and DECnet end-node look-a-like.
- Very good price/performance.
- Management are smart businessmen that have brought Convex along steadily. A competitor to watch out for in both technical and commercial markets.

HEWLETT-PACKARD:

- HP9000-8XX file server, based on RISC Precision Architecture and running HP-UX, is closely positioned against DIGITAL.
- Montgomery Securities claims "strong acceptance" of HP9000-8XX in commercial UNIX market in 1988.
- Active in enhancing UNIX with real-time extensions, NewWave GUI.
- Now owns Apollo, a high performance workstation vendor.
- OSF member.

IBM:

- AIX/370 due out late 1989.
- TP subsystems (such as CICS) are not expected until 1993-1994.
- Multiprocessor RISC platform (RT family) expected late 1989.
- Controlled 45% of total U.S. business computer revenues (proprietary and UNIX) in 1987.
- Will leverage OSF selection of AIX kernel in sales situations.
- OSF member.

NCR

- Tower series running Tower OS (UNIX) has been successful in retail and financial applications.
- Good low-end scalability.
- Rumored candidate for takeover.
- UNIX International member.

PYRAMID:

- Closest direct competitor to DIGITAL (high end commercial UNIX/RISC servers).
- Dual BSD and System V UNIX (OSx 4.4).
  - — Cache subsystem to improve throughput.
  - — Disk mirroring option.
  - — Disk striping.
  - — Virtual disk capability.
  - — Supports up to 4GB virtual memory per process.
  - — RPC facility.
- Pyrnet networking products include:
  - — SNA/3270 gateway.
  - — Cray channel interconnect.
- 85% of its systems used for commercial applications.
- 50% of its sales are to telecommunications industry.
- Installed base of over 1600 systems.
- Over 50% of sales are outside of the United States.
  - — Nixdorf accounted for 20% of all 1988 sales.
- UNIX International member.

SEQUENT:

- 80% of its systems used for commercial applications.
- Offers dual BSD 4.2 and System V R2 UNIX (Dynix) running on an Intel 80386-based multiprocessor platform.
- Signed original equipment manufacturer (OEM) agreement with UNISYS in January 1989.
- Joint venture with Matsushita to enter Japanese commercial market.

- OSF member.

SEQUOIA:

- Fault tolerant multiprocessor platform (Motorola 68030-based).
- Offers SVID issued compliant UNIX called Transaction Optimized Processing with UNIX (TOPIX).
- Operating system provides fault detection (error detection, self-checking, and protocol monitoring), fault isolation, and fault recovery.
- Applications designed for System V or BSD can run concurrently.
- Proprietary flat file system called Sequoia Transaction Oriented Record Manager (STORM).
- Turned first profit in seven year history during the fourth quarter of 1988.

STRATUS:

- Native fault tolerant UNIX expected late 1989; to be both POSIX and SVID compliant.
- Broadening scope from fault tolerance to "minicomputer based critical on-line application solutions" (1988 annual report).
- OEM sales about 40% of revenue (IBM 32%, Olivetti 8%). IBM resells XA/2000 as System/88.
- Member of both OSF and UNIX International (hedging strategy).

SUN:

- Offers workstations and (currently) low-end servers with wide range of applications that span technical and commercial (trader workstations).
- Joint project with AT&T (owns 20%) to improve System V.
- SPARC server strategy under review. Just announced a new line of powerful workstations and rumored to be working on multiprocessor SPARC servers.
- UNIX International member.

TANDEM:

- Also broadening scope to general OLTP.
- Offers UNIX running on non-fault tolerant LXN, targeted at small businesses–a new market.
- Working with MIPS to include RISC in future products. Gartner Group expects Tandem will first offer RISC with its Guardian OS.
- Key acquisitions strengthens production systems offering:
  - — Atalla Corp.: secured transactions.
  - — Integrated Technology: telecomm industry products.
  - — Ungermann-Bass: networking.

TOLERANT:

TBS

UNISYS:

- Has targeted commercial UNIX market as source of future growth, as its proprietary systems sales decline. Claimed in recent Wall Street Journal ad to be "#1 supplier of commercial UNIX." Recently acquired Convergent, one of its OEMs.

- Has added extensions to System V R3 to make it more attractive to commercial users:
  - Menu system.
  - Encryption.
  - Virtual address support.
  - File system tuning options (allocation of contiguous disk space, direct I/O support).
  - Real-time extensions (preemptible scheduling, asynch I/O).
- In strong position to influence System V TP standards.
- Actively pushing for a IEEE (POSIX) TP standards working group.
- Has moved quickly into market by reselling systems by NCR, Arix, Convergent, Computer Consoles.
- UNIX International member.

4 Opportunity Analysis

4.1 Open Systems Market Today

The open systems production environment today is portrayed by the industry forecasters as small and immature. While this is inaccurate, it is true that most of the current UNIX implemenations do not provide many of the capabilities necessary to support this environment. By exapmle:

- AT&T has been shipping production systems based on UNIX and Tuxedo/TP[1] for many years mostly into the telecom industry for applications such as 1-800 number conversation, telephone billing systems, and network maintenance.
- UNISYS, a provider of production systems, has just reported over $1 Billion in revenue derived from open system sales.
- NCR is incorporating UNIX systems into their retail solutions.

Other competitors such as IBM, Hewlett Packard, and Pyramid are heavily investing in open production systems. They bring new technologies such as multiprocessing, databases, and transaction monitors as well as marketing prowess and complete customer support. Unfortunately, DIGITAL's share of this market is neglibable.

4.2 Open Systems Market in 1992

By 1992 the total UNIX market is forecasted to reach $24 billion ISS (Initial System Sales). The open systems production market will be $13.6 billion, ISS. This is a conservative sizing measured at a point where the technology is not mature. Vendor interest is high with the focus driven by a group of customers led by governments, aerospace, education and telecomm. Those and a cross section of other industries are dictating that future purchases meet standards. With this comes a demand for the features found in production environments.

As open system vendors respond to the demand for production oriented customers, the market for open systems will grow. Some competitors are introducing technology, such as Tandem and Stratus, with continuous-operation computing, or Convex with high performance and availability. Others such as IBM, AT&T/Sun, HP/Apollo are investing heavily in software technology targeted for the commercial markets. And others such as Olivetti, NCR, and UNISYS are integrating production solutions for targeted markets such as banking and retail. Software vendors such as Oracle and Sybase are providing high performance databases. Open system solutions will drive even more demand, increasing the rate of growth.

Digital can exploit this opportunity by providing a new and complete production oriented open system rather than retrofitting new functionality into existing products. The window of opportunity exists between now and 1992.

4.3 How Does DIGITAL Win with OZIX

For Digital to win in the open systems production environment it must differentiate itself on how it delivers the solution. Delivery is a combination of third-party application integration, sales, field support and training. The following outlines basic programs for a successful introduction of OZIX.

- Marketing must properly position OZIX versus ULTRIX value, VMS, and open systems.
- The System Engineering groups should adapt the third-party solutions to Digital's system platforms such that Digital provides the best price/performance per application in the industry.

[1] Tuxedo/TP is a registered trademark of AT&T

- Sales needs the tools that correctly configure complete solutions. This involves good configuration tools and access to information on the applications available. This information should focus on the open systems production environment, the source comes from both transaction processing and open systems.
- Software Services must have the expertise to support sales providing technical depth on Digital and third-party products. To generate revenue they should also develop service packages for the open system production environment such as application integration, capacity planning, and customized applications.
- Field Service should be able to first service all of the Digital's open systems equipment without using VMS. The service tools need to be upgraded to provide diagnostic tools that match the production or mission-critical environment. New service offerings should be designed that match the production environment's need for 100% availability. An example would be a diagnostic tool that monitors the system against a pre-set error tolerance level that when reached automatically dials a service center and asks that diagnostics be run. This is done while the application is running.
- User oriented training should ship with the system. Comprehensive application development training that focuses on programming techniques, coding efficiencies, and programmer productivity should be available at FRS. The course should make full use of OS/SB's CASE environment, providing the customer the understanding of how our tools do improve efficiency. These courses should be tailored so they can be delivered either at a Digital facility or at a customer site.

5 Product Requirements

The following product requirements, based on market requirements, are essential for DIGITAL to be successful in the production environment open systems market. This section addresses the requirements for a project, code named OZIX, which is one product in a family of open systems products, called ULTRIX, produced by OS/SB engineering.

These requirements are targeted at the TP computing style, which is not the first release of the product but is a long term product goal. OZIX will be introduced (first release) into the compute and file server market where enhanced security and reliability are attractive features. The initial product release is designed to gain experience in the production system market and establish a verifiable track record for reliability and availability.

5.1 Standards

Industry standards (programming, user, and system interfaces, communication protocols, etc.) are what define systems as open systems. The standards appear below in priority order. Conflicts are resolved in favor of the higher ranking standard.

1. X/Open XPG4 branding
2. OSF AES (See Appendix B for complete list)
3. POSIX
4. ANSI
5. FIPS 151-1 (additions to POSIX)
6. ISO
7. SVID
8. UNIX defacto

NOTE: Details of exact standards conformance is to-be-supplied in a later revision of this document.

The following list details additional functionality either required or highly desired at OZIX first release:

- Required: additional OSF functionality as the specifications become available from the OSF
- Required: industry standard distributed name and time services
- Required: OZIX development language must enable DIGITAL to submit technologies to the OSF and to the various standards organizations. Therefore, the development language must be ANSI C or C++
- Required: Support for industry standard hardware interfaces, specifically SCSI and VME

5.2 Security and Authentication

OZIX is intended for use in both government and commercially secure sites. The worldwide trend toward more secure systems requires that OZIX be certified at DoD security level B2.

OZIX security must be designed to allow customers to modify or customize the security features to meet their specific installation and application requirements. OZIX security must also be designed to support various security models providing flexible security to user data. In addition, a distributed user identification and authorization mechanism is required for customers who must operate worldwide networks.

There is a direct relationship between system performance and security rating. The more stringent the security requirements the greater the penalty on system performance. By providing a scalable, modular security model customers do not pay a performance penalty for unwanted security features.

5.3 Interoperability

Open systems installations have equipment which span the range of size and processing power from small personal computers to large mainframes. Vendors providing systems into this environment must interoperate with systems from IBM, UNISYS, Sun, AT&T, Pyramid, Sequent, Tandem, Stratus, Convex, HP, NCR, DIGITAL, and others. These systems run a variety of operating systems, both proprietary and open. Operating systems include: MS-DOS, OS/2, MVS, AIX, VM, VMS, ULTRIX, System V, HP/UX, Sun/OS, A/UX, OSF/1, and other UNIX-based operating systems.

5.3.1 ULTRIX Interoperability

OZIX, as a member of the OS/SB ULTRIX product family enables customers to create portable applications that span DIGITAL's open systems implementations. It allows these applications to interoperate freely regardless of the underlying hardware architecture.

To provide a high level of compatibility and interoperability with other OS/SB open system products OZIX must provide:

- A common calling standard
- A common object file format
- Support for user-written UNIX-style device drivers

5.3.2 VMS Interoperability

The following list details functionality either required or highly desired for OZIX to maintain a high degree of interoperability with DIGITAL's VMS operating system.

- Required: DECnet Phase V (OSI)
- Required: Application Integration Architecture (AIA) components. The list of AIA components include: CDA, DECwindows, DECforms, GKS, PHIGS, SQL, DECprint, language RTL's, CMA, authentication services, DTTM, and math RTL's
- Highly desired: DIGITAL's distributed system services (DSS) which include distributed name services (DNS), distributed time services (DTS), distributed queue services (DQS)

5.3.3 Industry Interoperability

OZIX must be able to connect to and exchange data with DIGITAL-supplied operating systems (such as ULTRIX and VMS), other UNIX-style open systems, and with non-DIGITAL proprietary operating systems (such as IBM's MVS). A suite of communication protocols and access methods must be provided which allow information exchange in these various environments. The following lists the various protocols and access methods that must be supported for leadership interoperability capabilities in transaction processing:

- TCP/IP—required to interoperate in an open system environment
- NFS—required to interoperate in an open system environment
- X.25—European standard and is also gaining domestic popularity
- X.400—electronic mail interchange
- RPC—efficient protocol independent messaging mechanism

- ANSI SQL and Extended SQL— required for database interoperability
- LAT—required for support of character cell and 3270-like terminals and LAT-based printers
- OSI (GOSIP compliant)—it will eventually become the industry standard networking protocol. DIGITAL will gain credibility for innovation by leading the industry with an OSI implementation
- SNA, LU6.2, data stream, and 3270 gateways and host-based access routines – required for interoperability with IBM mainframes and large customer networks in a transaction processing environment

5.4 Performance

Performance can be a clear differentiator in the area of open systems. OZIX must perform at a level higher than that of any other OSF-compliant operating system, at the same security level, executing on identical hardware. Performance characterization and optimization tools must accurately predict peak load and performance bottlenecks. OZIX must also provide an effective self-tuning mechanism for maintaining optimum operating system and application performance. Self-tuning is vital when hardware failure occurs and the system must continue to perform, although in a degraded capacity. OZIX must have the ability to quickly and efficiently adjust itself when such a situation occurs.

Application performance analysis, optimization, and characterization tools are also important. End-users will see system performance as a function of their application. Application characterization tools are needed for estimating or predicting potential bottlenecks in the applications themselves.

OZIX market opportunities require support for a high-speed database and transaction monitor that supports the full range of DIGITAL's desktop strategy.

OZIX must support DECwindows on workstations, PC's and MAC's, character-cell terminals for data entry via local area transport (LAT), and 3270 terminals for data entry and interoperability with IBM systems.

OZIX must support standard debit/credit transaction rates that range from a minimum of 30 transactions per second (TPS) to a maximum of 400 TPS. I/O performance optimization features such as disk striping and shadowing are essential since TP-based applications are very I/O intensive.

NOTE: I/O rates to support transaction requirements are being studied and will be supplied with the next release of this document.

OZIX must support I/O connectivity for tasks required by the customer base. For the first release of OZIX, the requirements are in the 200-300 gigabyte range. By the mid-1990s, one Terabyte will be required to support the transaction services and databases. Other devices such as high-speed tape drives and extended memory or electronic storage devices must also be supported.

As the processing power of systems increases, network bandwidth increases as well. Demand for high-speed network connections must be satisfied by supporting leading-edge technologies such as FDDI. It is also important to support current technology connections such as Ethernet. Multiple path connections to both these networks is essential to assure high bandwidth as well as redundancy.

The peripheral connection strategy also requires that the hardware provides high-bandwidth connections to disks, tapes, and networks. Multiple paths to each must be maintained to support failover in case a single path is lost.

Application software that provides management of data on a system-wide basis is very important in the TP environment. Data management software must provide for automatic migration of frequently used files for quick access of media (electronic storage, solid state disk, etc.) while less frequently used files are moved to slower media or are archived for later retrieval. It is extremely important that individual files are automatically optimized for maximum performance without service interruption.

Backup and restore mechanisms must be improved over current technology. New and faster methods must be devised so that service to the end-user is not interrupted. Backup of user data must be possible in one shift (8 hours).

Memory requirements for the first release are in the range of 256MB as a minimum. For transaction services, 1-2GB are required.

5.5 Internationalization

OZIX must support a multioctet character set (MOCS) environment, available worldwide at first release. Base systems, utilities, and commands must support multibyte (1 to 4) characters, including data representation of dates, times, currency, numbers, and sequences collated in any language. OZIX must also support display of multilanguage characters within a single string. System informational messages must be isolated to enable easy local translation. System and user documentation must also be easily translatable.

5.6 Languages, CASE Tools, and Applications

At first release, OZIX must provide a suite of CASE tools and applications (developed by both DIGITAL and third parties) that address the needs of the target market. The quality and quantity of products available for OZIX must be expanded as quickly as possible to cover identified areas (TP) and unidentified areas of opportunity. Applications must be available in a very broad price range in the OZIX target areas.

CASE tools must be made available as soon as possible, preferably during field test. A strong CASE environment will attract both software developers and end-user development.

The proper CASE architecture includes tools that span the life cycle of the application from development concept to retirement. The CASE environment should be open to allow third party software developers to integrate their tools and flexible in methodology to allow the environment to adapt to the application developers work style. This will create an atmosphere that will encourage application developers to use OZIX as a development platform to develop new applications and to port existing applications.

First release target areas for applications and tools include business computing, servers, and CASE (both commercial and technical). Business computing systems require applications for decision support, screen management, database, and commercial CASE tools. Server products require database, communications, file services, and graphic libraries for workstation support.

In addition to these short-term areas, products must be developed for application and database prototyping, performance characterization for the operating system, networks, and applications and a complete set of traditional and 4th-generation languages. All languages and tools should adhere to the current standards if they exist (for example, C must be ANSI X3J11, Ada must be ANSI/MIL 1815A-1983 and FIPS 021-2, and so forth).

For a complete list of CASE tools and applications that are required for OZIX please refer to the OZIX Third-Party Applications Plan.

5.7 Third Party Support

A third party application software program is critical to the success of of DIGITAL's open software effort. OZIX will not be successful without the availability of third party application software at FRS. Customers want solutions today. This means application software and is an extremely important aspect of doing business in the 1990s in the open systems market.

As experienced recently by the shipments of our RISC-based workstation products versus original projections, it is not enough to just have *hot* hardware. As hardware becomes more of a commodity, interest in software becomes more acute. In many cases the application software decision has already been made, it is just a matter of the customer finding the best system available to run the application.

To ensure that applications are available, not just announce as intent, many milestones must be met in the near future. The following list outlines these steps:

- The first step is to identify and prioritize the products
- Second, establish the type of relationship with the vendor. This is a critical step in the process since it may dictate the level of DIGITAL supplied support that a third party software vendor will receive
- The last step is the actual acquisition of the software

Porting aids (documentation or software or both) must be developed to assist third party developers who do not qualify for the higher levels of DIGITAL support (aid from Product Marketing Groups) to port applications to OZIX-based platforms. In some cases porting aids alone will not suffice. In cases such as these perhaps some level of remedial support, through engineering or software services to be reviewed on a case-by-case basis, should be established.

5.8 System and Network Administration

Integrated system and network management/administration provides the foundation for many elements of the system. The following critical areas must be addressed:

- Ability to manage many machines or users with the fewest resources (low cost)
- Management of nonstop systems
- Resource scheduling (class scheduling)
- Print and batch queue management
- Asset (software) management
- Security management (system and network)

At a minimum, OZIX must provide an integrated system and network management/administration for all OS/SB-supplied operating systems and networks from the client (that is, workstation) or server. Systems and networks include ULTRIX/OSF, OZIX, OSI, and TCP/IP at first shipment.

As OZIX matures, it must provide a platform for management of other operating systems and networks that comprise the customers environment. OZIX system and network administration software must be designed in such a way that management of foreign systems is possible. Examples of operating systems that OZIX system administration should support include VMS, MVS, VM, OS/2, and other UNIX offerings. Examples of networks that OZIX network administration should support include OSI, TCP/IP, SNA, and X.25.

OZIX must also provide a distributed batch and print mechanism. Print and batch jobs must be accessible to both the user and administrator. Batch jobs should be executed on appropriate hardware automatically without user or operator intervention. The batch processor may need to understand job requirements which relate to memory requirements, number of processors needed (decomposed application), disk or file resource constraints, or hardware failures. Customers should be able to place their print jobs on any available printer.

Centralized software and online documentation distribution on a single distribution medium (CDROM) is essential. OZIX must provide a mechanism for management of the software in a distributed environment. Software licenses should not be tied to a single machine but should allow the software to be executed on any compatible system.

Software installation, updates and upgrades must allow for selective upgrades, be easy, and straightforward to perform and cannot interrupt operation of the running system. A highly desirable feature of OZIX is support for multiple concurrent versions of layered products or applications running simultaneously on a given system.

Today the time needed for installation of a DEC operating system can range from several hours to several days based on the operating system and the distribution device. Typically, this has not been a serious concern for customers in the technical and scientific markets. In the commercial market, however, customers will not accept long installation times. Installation must be as easy and transparent as possible.

To achieve short installation times and to make the installation as transparent as possible, OZIX must be shipped preinstalled to the customer. The preinstalled version of OZIX must be shipped in a standard configuration that will fit most customers. Therefore the predefined, standard OZIX system will be usable by the customer immediately after booting.

In addition to the standard, prebuilt installation, the customer should be given three options:

- Customization at the time of installation, adapted to the unique customer environment and provided by DIGITAL either on site or remotely
- Custom-built, turnkey installation loaded on the customer's system disk prior to shipment. This would be based on the customer's input, prior to system shipment (as is done with the VAX 8974 today)
- Customer performs any needed customization without DIGITAL's assistance

5.9 Availability and Reliability

A customer's perception of availability and reliability is based on the effective functioning of the entire system, which includes applications, layered products, hardware, and the operating system. Availability and reliability of OZIX will depend on a number of different aspects. Actual availability and reliability numbers are heavily dependent upon the specific hardware configuration. Expectations, however, for the production system market require a minimum of 99.5% availability. Applications requiring a higher availability require additional hardware.

A critical concern of open system, production environment customers is the reliability of the file system. OZIX must provide a robust, fault-tolerant, high-performance and recoverable file system.

To provide high availability, complete and thorough testing of the hardware and software is essential to ensure high reliability. OZIX must also provide fault detection and recovery mechanisms to prevent catastrophic system conditions or crashes. Fault management must be built into the base system. Hardware and software failures must be contained to the smallest unit possible and must not affect other systems in the network. In the event of a system crash (hardware or software) it is essential that the system provide a fast recovery mechanism. Full system recovery must take no longer than five minutes. OZIX must maintain an average of less than one crash per year. In addition, OZIX must be able to automatically detect, configure, and bring new devices on line, as well as remove devices from service if and when they reach a specified threshold. All of this must happen without service interruption.

Please refer to the OZIX Serviceability Requirements Documents for specific serviceability requirements.

5.10 Scalability

OZIX must provide near-linear performance (scalability) within a single (multiprocessor) system and across the supported range of hardware. Scalability across a distributed network of OZIX systems should be limited only by the bandwidth of the communications link.

OZIX must be able to support more reliable and higher speed hardware architectures with as little porting effort as possible. Although time-to-market is not the overriding consideration, technologies are changing at a rapid pace. OZIX must keep pace with technology to be competitive.

5.11 Usability

Customers desire systems which are usable. Interfaces such as those used for system and network management, and online documentation must be designed to fit the needs of the appropriate users. In addition, the interfaces must be tested to determine how usable they are.

Please refer to the OZIX Usability Plan for additional information.

5.12 Services

Services are essential to the success of products in the production systems market. Customers want and need to feel comfortable with their vendors and confident that all their service needs can be met by the vendor.

Please refer to the OZIX Customer Services Requirements Document for specific service requirements.

5.13 Documentation

Customers expect a high quality information set that is geared to both users and administrators. The information set must be concise and easy to use with emphasis on online presentation and delivery. It must be optimized for easy translation to other languages.

Please refer to the OZIX Master Documentation Plan for specific details.

5.14 Quality

The quality of the OZIX product must be maintained and proven to the customer through all components and subcomponents of the system. Quality and robustness of the product will not ensure success but must be maintained for OZIX to be successful. OZIX must be rigorously and fully tested at subcomponent, component, workstation, and network levels.

5.15 OZIX Positioning

DIGITAL has two primary operating system families. With the introduction of OZIX the ULTRIX family will be expanded to meet new customer demands in non-traditional open systems markets. ULTRIX and VMS target different markets, answer different customer needs, and offer the customer different capabilities. Both operating systems are critical to DIGITAL's success.

OZIX vis-a-vis ULTRIX

OZIX is complementary to the ULTRIX family. While ULTRIX is targeted at traditional UNIX markets, OZIX is aimed at markets where UNIX systems are not generally used today. OZIX is the product that allows DIGITAL to expand its opportunities in the open systems market by delivering high-performance, highly-secure, distributed systems to an international marketplace. Because OZIX is being built with the latest design and engineering methodologies, these capabilities can be fully integrated into the base system.

ULTRIX is DIGITAL's traditional UNIX product offering and will continue to better answer the needs of customers in these markets while remaining a low-cost solution. Those customers who required the added capability that OZIX offers will pay a higher price for OZIX than for ULTRIX.

OZIX vis-a-vis VMS

The VMS operating system is the most complete, functional operating system on the market today. It successfully addresses the needs of customers across a broad range of markets with its primary competition being IBM's MVS and AS/400 systems. The greatest distinction between OZIX and VMS is that OZIX meets the complete set of criteria for an open system. While VMS will implement POSIX, OZIX *is* a UNIX system and competes directly with other UNIX and open software production systems.

OZIX vis-a-vis Other Open System Offerings

Current open system offerings will have serious problems providing customer solutions for tomorrow. For example[1]:

- Bill Joy - "We see a lot of people having multiprocessors; we see a lot of people having distributed systems connecting lots of computers together, and, that's not the kind of environment that there was when UNIX was first defined so we want to redefine what UNIX is so that it really works."
- Professor Richard Rashid, Carnegie Mellon University - "UNIX no longer is easy to modify as it once was...B.S.D. contains more than 100 system calls and hundreds of system call options...the UNIX kernel has become a dumping ground for virtually every new feature or facility."

Known problems with current implementations include:

- Extensions require kernel modification
- Known security holes
- Distributed computing deficiencies
- Modularity and implementation problems

OZIX addresses each of these issues directly by providing a new modular implementation designed to be extensible and expandable to meet future operating system demands.

[1] Excerpts from "The Architecture of Future Operating Systems, by Richard W. Watson, Lawrence Livermore National Laboratory, Nov 1, 1988.

APPENDIX A

OZIX PRODUCT SUMMARY

A.1 Product Summary

The OZIX product is a member of the ULTRIX product family with significant enhancements for applications environments with enhanced security and reliability requirements in an open system are required. The OZIX design center is networked servers and platforms for high-performance transaction processing used in production environments. The product capabilities are developed over a series of releases and are composed of system elements from other DIGITAL groups (OS/SB engineering, SDT, TP software, database manager, compilers, tools, utilities), as well as selected elements from third party software suppliers.

The capability of the base system in both function and performance will extend significantly beyond that provided by other open systems implementations (for example, the OSF reference code). The level and quality of integration provided in this distributed computing environment will also differentiate DIGITAL's products from that of its competitors.

OZIX is an integral part of a complete family of OS/SB supplied distributed open software systems which includes worksystems, departmental timesharing systems, and servers. OS/SB supplied systems are carefully integrated into the total DIGITAL distributed computing environment which can include VAX/VMS systems and personal computers and connections to industry-wide systems. Other elements from DIGITAL's desktop strategy will be incorporated, as application focus dictates.

In order to successfully compete in the continuous-operation production environment, software and hardware together must have a proven track record, from an application perspective, of stability and reliability. To establish this record, the initial version of OZIX will be released in application areas less demanding than TP. The first TP capabilities will emerge in the second major release. The long-term goal of OZIX is to compete successfully for major TP business and attain clear leadership in the distributed production environment open systems market.

A.2 Major Development Areas

Development of OZIX emphasizes the following major areas:

- MIPSco- and EVAX-based processors
- V1.0 expected to ship on LASER/R4000 (Q4FY91)
- V2.0 expected to ship in Kestrel timeframe
- Production environment quality and reliability
- Pervasive, consistent, and configurable distributed security model, certified at DoD B2 level
- Meet all OS/SB familiness requirements
- Compliance with standards specified or required by our potential customer base, such as the Open Software Foundation (OSF), X/Open branding, POSIX, and so forth

- Superior system and application availability, fault tolerance, and data integrity
- Provide a high-performance platform for DIGITAL's TP and database products (DECxTP, DDTA, and QUARTZ)
- Incorporation of elements from third-party sources as a way of achieving and maintaining leadership
- Integrated system and network administration as a recognized product strength (ease of management)
- Leadership computing and I/O performance through innovation of base system technologies
- Available worldwide at FRS, developed to satisfy the requirements of the international business community
- Superior documentation
  - — Tailored for online presentation
  - — Structured to recognize the needs of different kinds of users
  - — Optimized for easy translation to other languages
- Licensing and distribution consistent with DIGITAL business requirements and industry direction
- Interoperability within the DIGITAL computing environment (VMS & AIA)
- Superior interoperability with IBM systems, especially for commercial applications

APPENDIX B

OSF AES SPECIFICATION (LEVEL A)

The following lists the qualifications to meet the OSF AES Specification (level A) guidelines.

- Operating system
  - IEEE POSIX standards (1003.1 - 1003.5)
  - NBS Federal Information Processing Standards (FIPS)
  - X/Open Common Application Environment (CAE)
- Languages
  - C: ANSI X3J11
  - FORTRAN: ANSI X3.9, ISO 1539-1980(E), FIPS 069-1
  - Pascal: ANSI X3J9, ISO 7185-1983, FIPS 109
  - Ada: ANSI/MIL 1815A-1983, FIPS 021-2
  - BASIC:

    Minimum Basic: ANSI X3.60-1978, FIPS 068-1
    Full Basic: ANSI X3.113-1987, FIPS 068-2
  - COBOL: ANSI X3.23-1985 High Level, FIPS 021-2
  - LISP: Common LISP, ANSI X3J13
- User Interface/Window Manager
  - X Window System Version 11, ANSI X3H3
  - Libraries: X Language Bindings, ANSI X3H3
  - Motif
- Graphics Libraries
  - GKS: ANSI X3.124-1985, FIPS 120
  - PHIGS: ANSI X3H3.1
- Networking Services
  - Selected ARPA/BSD services
  - TCP (MIL-STD-1778), IP (MIL-STD-1777), SMTP (MIL-STD-1781), TELNET (MIK-STD-1782), FTP (MIL-STD-1780)
  - Selected OSI protocols

- Database Management
  - SQL: ANSI X3.135-1986 (with 10/87 addendum), levels 1 & 2, FIPS 127

Internationalization Plan for OZIX V1.0

at Phase 0/1

Produced By:

Mike Oughton, Senior Product Manager, IED/R

Revision Information:

17-Jul-1989,	First Draft	Limited Review
26-Jul-1989,	Rev 1.0	Country translation input
14-Sep-1989,	Rev 1.1	DECwest Review
16-Oct-1989,	Rev 1.2	Final - OZIX Phase 0/1 Closure

For approval by:

Mike Parker, Corporate Product Manager
John Gilbert, Engineering Manager
Bob Dray, ISSG Group Manager

Corporate, International Engineering Development (IED) and Asian Based Software Systems (ABSS) :

Terry Morris, OZIX Product Manager
Cathy Richardson, OZIX layered products Product Manager
Benn Schreiber, OZIX Project Manager
Claire Cockcroft, OZIX I18N Program manager
Jim Jackson, OZIX Documentation Manager
Eric Getsinger, OZIX Documentation I18N DRI
Tim Greenwood, ABSS
Ron Brender, ABSS
Mike Feldman, IED/Israel
Jürgen Bettels, IE Architecture
Eamon MacDermott, IED/R
Martyn Kee, IE/CUP
Andy Vowles, Arabic Products, IED.
Pete Smith, OSCR
Omur Tasar, OS/SB Enterprise Management.
Jim Mills, IED Manager (Staff Distribution)

GIA

Vince Bastiani, GIA, Systems Strategy Manager
F. Avery Bishop, JRD, Japan
Bill Fulton, Australia.
David Argue, Canada.

LEG Managers (Europe)

AUSTRIA	-Manfred Zoempfenning
BELGIUM	-Christine Robeir
DENMARK	-Erik P. Andersen
FINLAND	-Anja Pentikainen
FRANCE	-Denis Rabourdin
GERMANY	-Klaus Steiger
ISRAEL	-Aharon Goldman
ITALY	-Michele Pennisi
NETHERLANDS	-Bob Krijger
NORWAY	-Olav Gude Gudesen
PORTUGAL	-Edmundo Silva
SPAIN	-Pilar Peces
SWEDEN	-Per-Ola Niblaeus
SWITZERLAND	-Joachim Goehring
UK	-Richard Buxton

Area Marketing (Europe)

Michel Filthuth, EHQ, European OZIX Marketing Manager
David Petraitis, EHQ, Open Systems Software Portfolio Manager
Jean-Claude Monney, EHQ, European Open Systems Program Manager
Emannuel Fizzarotti, EHQ, European Marketing (Engineering Liaison)

Quality

John Prestige, SQG

CSSE

LeeAnn Stivers, OZIX CSSE Product Manager
Thomas Siebold, OZIX CSSE Maintainability Engineer

Others

LuAnn Van de Pas, Project Leader, RISC ANSI-C compiler
Chuck Piper, OSCR CASE Product Manager

Table of Contents

List of Figures

APPENDICES

1 PURPOSE OF THE PLAN

The Internationalization Plan (I18N Plan) forms the commitment from International Engineering Development (IED) to:

Provide the Corporate Product Team with consulting services

Work with the Corporate Product Team when applicable, to provide the development for, and delivery of, Product Variants

Coordinate localization work at area and country levels for Europe and General International Area (GIA).

The I18N Plan:

Identifies those dependencies that IED and the countries have, on the Corporate Product Team, to meet these commitments

Provides the method for measuring progress of the internationalization effort.

The I18N Plan has four main sections, Overview, International Requirements, Specifications and Project Plan. These should be seen as supplementing the following Corporate Phase Review Process documents:

The OZIX Business Plan

The OZIX Market and Product Requirements Document

The OZIX Product Specification(s)

The OZIX Development Plan(s)

Each section within the Internationalization Plan aims to add value to these Corporate Phase Review documents from an international perspective.

2 OVERVIEW

OZIX is DIGITAL's planned new Open Systems operating environment for reliable, high-performance production systems. OZIX is in Phase 0/1 at this time, with combined Phase 0/1 closure scheduled for 15th November 1989.

The specific purpose of this Internationalization Plan for OZIX is to form a repository for:

1. International functionality requirements
2. Translatability requirements
3. Translation Plans (as appropriate)

This plan will be updated prior to OZIX Phase 0/1 closure.

The primary requirements on OZIX with its bundled and other related components are:

- Multilingual international functionality, including support for Asian and Semitic languages and culture
- Translatability: separation of text from code, and use of a facility capable of handling multilingual messaging

Decisions to translate individual components (or parts of components) are business decisions that must be made between the Product Business Unit (PBU), Area Marketing and Country Marketing. Those decisions and an overall localization plan (if applicable) will be documented here. Individual country Localization Plans that are associated with the overall Plan will be produced when appropriate and included in Appendix A of this Plan.

2.1 BUSINESS PLANNING

This Internationalization Plan in its current Rev 1.0 form, provides the information to Area and Country marketing to enable a first-pass translation decision to be made. Translation decisions made at this stage are a commitment to apply the required resources that are needed to investigate further the business justifications

and the translation workloads involved.

OZIX Product Variant Business Planning data is required in support of translation decisions, and will be added to this document as appropriate for OZIX Phase 1 closure.

2.2 PRODUCT DESCRIPTION

OZIX is the code name for a project within OS/SB to develop a high-performance, secure, high-availability, production system for the Open Systems market. OZIX is a UNIX[1]-compatible system aimed at high-end server platforms. As such, it competes against other vendors' UNIX-based production systems. OZIX is complementary to ULTRIX, which is aimed at the workstation and low-end server platforms. OZIX is differentiated from VMS by its support of the Open Systems operating environment.

OZIX is a strategic product for DIGITAL. Over time, it will encompass many other DIGITAL and Third Party applications and services. The design centre for OZIX is networked servers and platforms for the class of high-performance transaction processing typically used in production environments. Therefore, product capabilities will be developed over a series of releases to achieve clear leadership in this particular environment..

In order to establish a track record of reliability and stability, Version 1.0 of OZIX will be released in application areas less demanding than transaction processing.

A largely unexploited market opportunity for DIGITAL exists in the Asian marketplace. OZIX is therefore placing particular emphasis on targetting this market, and will therefore be required to support Asian language translations, and Asian cultural conventions in addition to its European and Semitic capabilities.

The end-users for OZIX V1.0 systems as delivered will be primarily technical - including for example, MIS departments in multinational companies. OZIX must therefore allow for concurrent multilingual and multicultural application development and deployment.

The OZIX vision is to evolve toward a goal of reducing the current dependence on highly-trained system managers and in-house MIS departments for purposes of system management and application deployment. To achieve this goal in a multinational environment will require an increasing depth of local-language translation, in addition to incorporating the ease-of-use facilities envisioned post-V1.0 by the OZIX product team.

2.3 FINANCIAL SUMMARY

IED has made provision in the FY90 Long Range Plans for consultancy, planning and architectural activities connected with OS/SB products. OZIX has the highest priority for these activities. It is somewhat premature to estimate the specific investment required by IED in OZIX until Phase 1 closure. However, at a first approximation, $130k has been reserved out of IED/R funding for OZIX consultancy and planning activity. Coordination of any European language translation activities would be incremental to this approximate figure.

ABSS (JRD) have estimated and allocated two engineers for OZIX localization work for all Asian languages.

The return on this investment will occur in terms of:

- Increased market opportunity - particularly in the Asian markets
- Reduced development costs in terms of international functionality for OZIX layered applications (both DIGITAL and Third Party)

3 INTERNATIONAL REQUIREMENTS

This section supplements the Corporate Product Requirements document for OZIX.

Essential background information about the design of hardware, software and user information for products selling worldwide can be found in the International Engineering Reference Set "Producing International Products" Rev. B (also known as the PIP Handbook; refer to Appendix A). It provides much essential, relevant information in an easy-to-read format.

[1] *UNIX is a trademark of AT&T.*

This Internationalization Plan aims NOT to duplicate information contained in the PIP Handbook. The goal of the Plan is to highlight any additional requirements and suggestions that are specific to OZIX, in order to make OZIX compliant with DEC STD. 066-3 (Policy for Designing Products for All Countries Designated as Strategic Markets), and to enable it to support the language and cultural requirements necessary for it to be viable in the Asian marketplace.

3.1 REQUIREMENTS FOR AN INTERNATIONAL PRODUCT

An International Product is one that allows international and multi-national customers to interact with the system using an individual user choice of local language and cultural conventions. A truly International Product can be localized easily. It will have been designed for the international market according to the standards and guidelines laid down in the International Product Model.

International functionality at the base-system level is a key requirement for the success of the OZIX operating environment. OZIX will form the basis of distributed systems, thus the concept of "International Systemness" is important. "International Systemness" refers to multiple products, each with multiple-language user-interfaces, and capable of working together on multiple systems in a network.

If OZIX base-system support for multilingual capabilities, for language switching and cultural preferences and for the code-set requirements of European, Asian and Semitic languages are built in from inception, it will be straightforward to provide applications with the mechanisms to allow users to interact with OZIX systems using their choice of native language and cultural conventions. OZIX will therefore be attractive both to monolingual and multilingual/multinational customers.

By building the highest possible level of international functionality into OZIX from the beginning, it is intended that application development and deployment for multi-national markets will be both easier and cheaper. OZIX will therefore provide an attractive platform for DIGITAL, Third Party and Customer applications development.

3.1.1 Software

The primary requirements for the OZIX V1.0 system are discussed on a component-by-component basis later in this Plan. The overall International Functionality and Translatability requirements for the OZIX system and its components are to:

1. **Be code-set independent/neutral.**
   - Initially, OZIX will have to support a number of code sets, some of which are not yet fully defined. The OZIX system and components must therefore have the potential to support *any* code-set (including MOCS) whilst retaining the appropriate writing directions.
   - Code-set independence is required in the Base System right up to the level of terminal services and some other external communications components, where device-specific code-set transformations will be required.

     Note: Refer to the document "Digital's Text Model for the Future - What Should It Be?" (Contact Jürgen Bettels, GPSDCC::JBETTELS) for the approach recommended by IED to achieve code-set neutrality (Compound String).
2. **Separate all program, user interface material and message text from functional code, and utilize a multilingual-messaging capability and language-switching facility.**
   - For components with a DECwindows/UWS user interface, this means the use of UIL and XNLS.
   - For non-DECwindows components, X/Open (XPG4) NLS-compliant messaging must be used.
3. **Conform to, or supercede existing Open Systems Internationalization standards.**
   - X/Open XPG4 "NLS" (Natural Language System) will be under review during the development of OZIX, and will include standards for Asian language internationalization.
   - XNLS V1.0 will be avaiable throughout OZIX development timescales for DECwindows-based applications. XNLS V2.0 is planned to be available in time for OZIX and this will extend support of

language switching and cultural preference switching to meet the needs of Asian and Semitic languages.

4. **Follow the International Product Model (ABCD model) for packaging the product.**

   - See the PIP Reference Set for a description of the ABCD model, and later in this document for detailed additional requirements regarding on OZIX components.

5. **Ensure that all on-line documentation and Help components are capable of supporting multiple, concurrent languages on the same system, and are structured in such a way as to allow for partial translation.**

   - OZIX must support Asian and Semitic languages and provide for the mixed writing directions associated with these languages.
   - OZIX must provide a language-switching mechanism for on-line documentation and online Help, in order to support the needs of a multilingual and multinational/multilingual user community.
   - Design of the documentation must take into account the need to separate and identify documentation modules by reference to the target end-user, to allow for partial translations throughout the life of the product.

6. **Third-party buy-ins and DIGITAL layered products bundled with OZIX systems also need to conform to these overall requirements.**

   Note: Attention is drawn to the Digital APA Message Text Services. These services will support MOCS with Compound Strings and may therefore be helpful in meeting requirements 1 & 2 , above.

   Note: XNLS V2.0 will provide language- and cultural-preference databases. DIGITAL layered products and Third Party buy-ins will be required (or at least strongly urged) to utilize XNLS services and profiles.

The OZIX components are discussed in more detail below.

3.1.1.1 Commands and Libraries

C run-time library (C RTL)

The emerging ANSI C Standard defines functions for wide character and special locale support. The C RTL should meet these standards.

Other Libraries

Whilst there is no requirement for translation of languages and compilers, code-set manipulation libraries are required for all programming languages. Wide character and mixed-direction string-handling is required in compiler RTLs.

Note: ABSS programming language datatype-compatibility requirements with existing products should be addressed in an ABSS-specific "C" component.

Note: IED is producing a Compound String manipulation library (CSlib) which is intended to provide the code-set manipulation libraries in OZIX referred to above. Contact Cliff Evans on VOGON::EVANS for details.

Commands and Utilities (OSF and BSD)

The commands and utilities form a user interface for command-line driven operations and for character-cell devices, and as such must be translatable and able to interoperate with the international functionality of the OZIX base system.

Commands and utilities shipped with OZIX must be structured for translatability. (Implement XPG-compliant messaging on all commands and utilities included with OZIX.)

Commands and utilities shipped with OZIX must also be code-set independent to allow for interfacing with the base system and for the handling of named objects that are named independently of the code set (for example, file names that are named in Japanese, and so on.)

Implementation-Independent Kernel Interface (IIKI)

No special requirements.

Printing (DECprint)[1]

Basic minimum fonts for all code-sets able to be created within OZIX should be provided within the A-component. As a minimum, it should be possible to read the content of a printed document even if the required typestyle/size of font is not available. Additional sizes and styles of fonts for non-Latin languages may be provided separately from the A component.

The printing function clearly needs to be able to support multi-byte character sets, to interpret the code-set neutrality of the base system and to convert this into a device-specific series of glyphs. The use of a non-readable fallback character (for instance, a reversed question mark, or block character) should be avoided wherever possible.

Mail Services[2]

The mail transport systems need to be able to transport multiple-language documents without loss of content and/or specific right-to-left or left-to-right directionality. It is suggested that this requirement be met by allowing the mailing of compound documents.

In addition to this specific requirement on the transport system, the Mail User Agents need to comply with all the overall OZIX internationalization requirements listed in 3.1.1.

Accounting

Whilst it is understood that accounting will be limited in OZIX V1.0, this component is nevertheless required to be code-set independent, and must utilize a language-selectable messaging facility to allow for translatability (including Asian and Semitic language translatability) .

3.1.1.2 Development Environment

Linker (ld)

No special requirements.

Librarian (ar)

The librarian should be code-set neutral except where it collides with current standards. It is understood that library names are restricted by convention (and the need to interoperate) to 15 ASCII characters. Librarian messaging should be separated from code to enable possible future translation.

1 *DECprint is effectively a layered product, however it is a bundled OZIX component.*

2 *Mail Services is effectively a layered product, however it is a bundled OZIX component.*

Linker Related Utilities (ldutils)

No special requirements

Message Text/Status Value (msgval)

This links message status with the messaging mechanism. The messaging mechanism needs to comply with XPG4, and also to support Asian language translations and mixed directionality of text strings in messages (for Hebrew and Arabic mixed right-to-left messages).

3.1.1.3 Diagnostic Monitor

The diagnostic monitor is the controlling software for hardware diagnostic tests, and also controls the system exercisers, disk formatters, and so on. Whilst DIGITAL Field Service is the primary user of the diagnostic monitor, it may also be used by third-party personnel responsible for maintainance, or by customers. Developers of the diagnostic monitor are following the same philosophy as for system management, that is, eventual migration away from the sophisticated user and toward the naive user. The user interface is being designed for flexibility, allowing different levels of interaction. It is believed that there ultimately there will be a potential translation requirement for the diagnostic monitor. Therefore all of the overall OZIX internationalization requirements apply to the design of the monitor.

3.1.1.4 System Exerciser

In addition to the overall OZIX international product requirements, the system exerciser must be designed to allow for its use on any localized OZIX components. SQM are requiring regression tests both on base and translated Product Variants. At a minimum, this means allowing for easy translation of any test scripts so that they may correspond with translated messages/displays. Ideally, the system exerciser should be entirely language-independent, making translation of test scripts for use on translated variants of OZIX components unnecessary.

3.1.1.5 OZIX Executive

Just as it must be possible to use local-language file names, with the appropriate code set(s) and writing direction(s), creation and use of local-language system identifiers must also be provided for, in order to cover such items as device names, process names, memory segment names and so on.

Boot/Recovery

No special requirements.

Subsystem Debug Server

No special requirements.

Virtual Nub

No special requirements.

Process

No special requirements.

Memory Management

No special requirements.

OSF Application Program Interface (API)

The interface must allow for multiple entry-point access to code-set independent mechanisms. (in particular, Compound String with MOCS and conventional 8-bit ISO-Latin1 APIs).

FIFO subsystem

No special requirements.

Security Support, Audit, and Access Validation Subsystem

The security model breaks the normal international system model because there is a need to access trusted messages (these are most likely to be written in English, to avoid dependencies on translation activities). It should be possible, however, for the audit file to also fetch an untrusted or "claimed" message part from local-language files that are outside the trusted computing base.

Likewise, file headers on printer output should be capable of containing a trusted header (for example, "TOP SECRET"), along with a "claimed" header in the local language.

It is not clear how a member of personnel who is responsible for security will interact with the OZIX security system (it is understood that some secure systems rigidly specify ASCII terminals). However, mechanisms should be provided to allow less rigidly-specified application(s) for use by security personnel to support input and display of text in the user's own language (including Asian and Semitic languages).

3.1.1.6 File System

File Name System

Any name that is visible to the user should be expressible in the local language (including Asian and Hebrew languages). Local-language support may require a format which allows for a mix of different character sets, as well as providing for different writing directions.

It is interesting to note that the Apple Corporation are apparently planning the use of graphics and voice for use as file identifiers. The use of Compound String in the file naming system would eventually allow this functionality to be a part of OZIX in addition to providing character-set independence and identification of the appropriate writing direction.

Storage Management Services

These services have no user interface of their own, rather the interface operations are achieved through the use of commands and utilities, and by means of system management. These services do however need to be able to interface with multi-lingual file names.

Networking

There are potential collisions between the existing standards and the goal of complete international functionality in the network area. Where these collisions occur, the standards must be adhered to until they change. However, documented workarounds (or alternative transports) must be provided to allow the transmission of multi-language data without loss of information.

Internet

Internet development must adhere to current standards, and must also recognize and document weaknesses in international functionality (for example, where 7/8-bit transmission is mandated).

Socket and XTI

No special requirements at this time.

NFS

NFS interprets file and directory names and must therefore be aware of code-sets in order to work compatibly with non-OZIX based systems.

Kerberos Authentication Server

It is planned for the server to be taken from the Athena development. This component must support all the the overall OZIX international functionality requirements listed in section 3.1.1.

Kerberos incorporates DES Encryption which has been restricted to sale within the U.S. by the U.S. Department of Commerce. However, it is understood that ULTRIX has permission to ship Kerberos worldwide.

3.1.1.7 Terminals, I/O Configurations, Network Devices and PIPE

Terminal Support

It is in the area of terminal support that code-set independence must, most obviously, be transformed into device dependence, with associated loss of information. The terminal-support software must be able to carry out the selection of character-set-specific modules in order to provide a selection that is appropriate to the connected device.

There is a requirement for a fallback mechanism to handle coded information that cannot be displayed by the connected device. This mechanism may display an appropriate (unusual) character from the device's repertoire (reverse question marks and block characters have been used in the past).

Whilst it is not required to support conversions to or from 7-bit National Replacement Character sets (NRCs), terminal-driver support should be provided in the A Component of the product for existing DEC 7-bit NRCs, the ISO 8859 family of character sets, plus the Asian[1], DEC7bit-Hebrew[2] and ASMO-449 (7-bit), ASMO-663 (8-bit), and ASMO-708 (8-bit Latin/Arabic) arabic[3] terminals.

(If existing UNIX application support is required to be maintained, other or new application use of NRCs should be actively discouraged.)
Any natural-language text messages must be separated from the code, and a multilingual messaging-utility must be utilized to allow for the potential translation of LAT.

The OZIX implementation of LAT needs to be code-set independent, must separate text from code in multilingual messaging catalogues, and must conform to current standards.

1 *Refer to Producing International Products Software Handbook. Appendix A.1.2 for details of existing Digital Asian terminals and code-sets. More information is available from ABSS. Contact Tim Greenwood ABSZK::GREENWOOD*

2 *ISO 8859 includes ISO 8859-8 Hebrew. For more information on Digital Hebrew terminals and character sets contact Aharon Goldman (TAVENG::GOLDMAN)*

3 *For more detail on Digital Arabic terminals contact Andy Vowles (VOGON::VOWLES)*

Note: For Asian languages, the following workarounds are planned:

One problem occurs during on-line font loading. The OZIX implementation of the terminal server plans to use a different class of terminal to workaround the xon/xoff problem.

The second problem occurs with the use of the autoforward feature between sessions. The basic workaround is to disable this feature when multibyte character-sets are being used.

I/O Configuration

See File System requirements for naming requirements. Any user-interface-to-I/O configuration is likely to be part of the system-management function, but the configuration must in any case be translatable and use a multilingual messaging facility.

PIPE

As object names are the province of the file system, there are no special requirements for PIPE.

CSMA/CD

No special requirements.

3.1.1.8 Mass Storage

There are no special requirements on the mass-storage structure, container, shadowing, striping, T/MSCP, CI, or the SCSI port and disk, other than the need to deal with code-set-independent object names and to conform with the standards and guidelines laid down in the Producing International Products Reference Set, Rev B.

3.1.1.9 RPC

RPC marshals string-arguments and therefore needs to support multiple character-sets and all appropriate writing directions. Any messages should be separated from code and must utilize a multilingual-messaging facility.

3.1.1.10 System Admininistration and Network Management

Of all the components of the OZIX system, this is the most likely potential candidate for eventual translation.

The OZIX vision incorporates the strategy of reducing the skill-level required to undertake system management and system-related administrative functions. Over time, if this approach is successful, naive users will undertake an increasing number of system-management responsibilities with the aid of easy-to-use tools. Translation of the user interface is therefore an obvious key to success in non-English speaking markets.

System administration and network management must therefore closely conform to all the overall OZIX international functionality requirements, where this does not conflict with specific ISO standards (for example, those relating to OSI/DECnet).

Additionally, because the migration from the technically-knowledgeable system-manager to the naive user will occur over a period of time rather than immediately, there needs to be a method of profiling users and user information. This profiling should be related to the configuration of system-management tasks by individual users. This will enable translation decisions that relate to material that falls within system management activities to be selective in relation to the spectrum of user-types (expert-to-naive). It will also allow consistent presentation of the appropriate local language to the user (by means of the appropriate user-configuration in terms of restricted access to the more technical and/or untranslated activities).

3.1.1.11 System Installation

There is a requirement for the installation mechanisms to install multiple language versions of the same product on the same system.

There is also a need for a mechanism to recognize the installed languages of products on a system, and this will require updating at system installation and also at additional language installation.

Installation dialogues should be translatable, be capable of containing multiple-language dialogues and should utilize a language-selection mechanism for selection of the presentation language.

3.1.1.12 Performance

The performance components consist of:

- PIXIE performance monitor
- Systems activity reporting tools
- PROF and GPROF performance displays
- Instrumentation

Whilst it is currently unlikely that these components will be considered as candidates for translation, they nevertheless should be code-set independent, and their design must separate messages from code in a multilingual messaging facility, in order to allow for any future requirements.

3.1.1.13 Layered Products Associated with OZIX

To be a fully International Product, OZIX has special requirements that must be satisfied by its bundled layered products. Where these products will be following the OZIX Phase Review Process, or are key to the success of OZIX, they are referenced in this document.

LMF

LMF on OZIX needs to recognize that there will be multiple (language) UIs for application products and that there may be marketing or business requirements for these UIs to be licensed separately.

DECwindows/UWS

DECwindows will provide the mechanism for the separation of form and function within the OZIX operating environment, and DECwindows should be the choice of user interface for applications requiring multiple local-language presentation.

DECwindows with XNLS will provide considerable support for multilingual user-interfaces and for culturally-sensitive interfaces.

There is also an additional requirement for DECwindows to support non-ISO-Latin 1 code-sets.

The key to meeting this latter requirement is the availability and acceptance of a text-handling model. A proposal for the use of Compound String is discussed in the document "Digital's Text Model for the Future - What should it be ?" by IED and ABSS (see Appendix A of this Plan).

OZIX has a requirement on DECwindows, UWS, and OSF/Motif next versions for support of multilingual, multiple character-set, and multi-direction text processing, as well as support for right-to-left UI geometry accross the whole widget set, and support of associated applications.

CASE Tools

Although CASE tools are limited for V1.0 of OZIX, they include the programming languages ANSI-C and

C++. However, within the languages, there is a need to support datatypes for MOCS and to access Compound String services.

All compilers bundled for shipment with OZIX must provide support for the development of International applications. This means support for multibyte wide character handling in source code comments, strings, literals and character constants. OZIX programming languages must provide access to the use of compound string within user programs. All development tools (compilers, loaders and debuggers) must have compound string handling capability.

Other OZIX Layered Products

Programmer Productivity Tools

Japan has in the past produced product-variants of portions of programmer productivity tools (such as LSE). Generally, programmer productivity tools need to be capable of supporting development of non-English software. Asian markets may also have some requirement for Asian language user-interface and documentation. Contact Tim Greenwood on ABSZK::GREENWOOD for details.

Today, programmers in Europe typically prefer to work in English because technical terms are more consistent and more readily understood in English.

Data Dictionary, RdbStar, DECforms

These products need to evolve to support MOCS by means of Compound String (refer to individual product Internationalization Plans - these are referenced in Appendix A of this Plan).

Office Products.

Today, most of the translation effort and most of the additional business has been gained with local-language variants of office products. Any DIGITAL-developed office product will have its own Internationalization Plan. (Contact IED on VOGON::I18N_PLANS for copies of existing Internationalization Plans that cover specific products.)

Third-party Buyins

Third-party buyins should conform to the overall international requirements for OZIX as stated in paragraph 3.1.1.

As many international issues can arise at the pre-contractual stage with Third Parties, it is generally beneficial to involve IED at an early stage.

3.1.2 Documentation

3.1.2.1 History

In the past, translation of documentation into different languages has been done on a selective basis, the extent of the translation differing according to the needs of the marketplace. This has typically involved production of local-language, naive end-user documentation, whilst the more technical documentation is left in English.

Problems have been experienced with this approach in the areas of packaging, getting the translated documentation to the end-user, and meeting the customer's requirement that they should be able to customize documentation to meet their individual needs.

Country engineering and marketing groups are now proposing to address these problems by NOT translating Corporate documentation, on the grounds that it is too expensive in terms of both translation effort and to the customer. It is argued that currently translated Corporate documentation does not reach the hands of the naive end-user,. It also leaves the system manager to cope with a mixed-language documentation set, when the re-

quirements for this rôle is a knowledge of the English language. Keeping translated documentation in step with releases of new versions also involves the Local Engineering Groups (LEGs) with continued expense after the production of the original documentation set.

For documentation other than that connected with Corporate products, the Country Engineering groups have taken the initiative to write separate books locally. These books are largely version-independent, and they are packaged and sold separately. The French Language Group (France, Belgium, Canada and Switzerland) have already followed this approach. (However, the needs of the broad range of customers for some degree of customization of documentation is only partly resolved by this approach).

The OZIX documentation strategy addresses the need for partial translation as shown below. There will be no need for locally-written books to be produced for the OZIX system products.

3.1.2.2 OZIX Documentation

The proposed focus for OZIX documentation is ON-LINE documentation, envisioned as the source of data for an on-line "Book Reader", for hardcopy documentation, for task-oriented Help and also for context-sensitive Help. Whilst this approach has clear advantages in terms of distribution, it raises some challenges in a multilingual environment where only naive end-user documentation is wanted in the local language, and where there is a need to minimize all translation costs.

In order for OZIX to be a fully-international product, OZIX documentation must address these challenges.

Specific requirements are:

- The tools used for on-line documentation must allow for creation and display of European, Asian and Semitic languages and must make provision for the appropriate writing directions.
- Multiple-language versions of on-line documentation must be installable on the same system. A mechanism must be provided for the user to select a primary and secondary 'preferred language' for the presentation of on-line documentation. This could be the same mechanism that the user would employ to select and set the preferred language for the user-interface. In the case of the preferred language not being available, either the default language (US-English) or a choice of the languages that ARE available on the system should be offered.
- In order to allow for partial translations (translation of naive end-user documentation only) to minimize translation costs and to retain an appearance of uniform documentation, documentation should be designed and written in a modular form by category(ies) of expected end-user(s).
- When a positive translation decision is made, tools to separate, edit, (retranslate) and re-combine documentation for translation from or with the documentation source-database must be provided, at least for DIGITAL Local Engineering/Translation Groups
- When a positive translation decision is made, tools to produce hardcopy documentation from the documentation source-database must be capable of producing translated, hardcopy documentation in European, Asian and Semitic languages. These tools must be available to the DIGITAL Local Engineering Groups
- System-manager configured user-profiling should be considered, in order to facilitate presentation of appropriate levels of technical detail on a user-by-user basis. This would also help to maintain the appearance for naive end-users of a consistent language for the documentation .
- Hardcopy documentation packaging should allow for separate, clear, concise and (ideally) version-independent, naive end-user documentation.
- The English OZIX documentation should be written for people who have English as their second language. (See The Producing International Products Reference Set, Rev B, for details.)

  Note: Although it is not a requirement of Internationalizatio, feedback from the Country Product Managers indicates that it would be very beneficial if these same tools were made available to the customer to allow the customer to make modification and additions to OZIX documentation.

3.1.3 Help

OZIX system-based task-oriented help and context-sensitive help is envisioned as coming from the same source database as the on-line documentation. The same internationalization requirements apply to on-line help as for on-line documentation.

3.1.4 CBI

IED is not currently aware of any plans to produce a CBI (Computer-Based Instruction) module for the OZIX system.

3.1.5 Hardware

Not applicable.

3.1.6 Packaging

OZIX and bundled components must follow the International Product Model (Refer to the Producing International Products Reference Set, Rev B, for details). OZIX packaging should allow for simplicity in ordering multi-lingual systems and must allow for user interfaces in additional language to be added post-FCS of the OZIX base product. International functionality components such as XNLS, with its multitude of built-in cultural databases and a minimum subset of fonts, need to be packaged with the base product for shipment worldwide. Translated "B Components" such as DECwindows and/or more complete local-language font sets may be packaged separately if required.

3.1.7 Testing

The test plan should include internal subsidiary testing of the international functionality as a minimum (for example, JRD of Japan, Israel and selected European sites). Due to the newness of some of the international functionality and the need to have applications to use it, it is recognized that it may be difficult to obtain effective external field test sites.

External multinational sites outside the US are however strongly encouraged.

3.1.8 Licensing

OZIX licensing should allow for simplicity in ordering multilingual systems and should allow for additional language user-interfaces to be added, and possibly separately licensed. This last decision is dependent on marketing requirements.

3.1.9 SPD/SSA

To be determined.

3.2 REQUIREMENTS FOR PRODUCT VARIANTS

A table is given overleaf showing countries on the top axis and components on the side axis. Initial country translation decisions are shown against these axes as explained in the key.

Fig 1. OZIX Component Translation Decisions Table

Key:
T = Translation of UI & DOC component required for V1.0
P = Partial translation of UI & DOC component required for V1.0
D = Defer translation decision until a later release
. = No translation required
? = No translation decision available

IED's *TRANSLATION* recommendation for Europe is similarly shown in the column marked "IE"

Language Groups:

UK = British English	GY = German	FR = French
SP = Spanish	IT = Italian	SW = Swedish
NO = Norwegian	FN = Finnish	DK = Danish/Icelandic
PO = Portuguese	NL = Dutch/Flemish	IS = Hebrew
AR = Arabic	JA = Japanese	SC = Simplified Chinese
TC = Traditional Chinese	KR = Korean	TH = Thai

			Language Group														
Class	IE	Component	UK	GY	FR	SP	IT	SW	NO	DK	FN	PO	NL	IS	AR	JA	SC	TC	KR	TH
Cmnds & Libs	.	C RTL	?	?	?	?	.	?	?	?	?	?	.	.	?	P	D	P	D	D
	.	Oth. Libs	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	D	Cds & Util	?	?	?	?	.	?	?	?	?	?	.	D	?	T	D	P	P	D
	.	IIKI	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
	D	DECprint	?	?	?	?	.	?	?	?	?	?	.	D	?	P	D	D	D	D
	D	Mail Svcs	?	?	?	?	.	?	?	?	?	?	.	D	?	T	D	P	P	D
	D	Accounting	?	?	?	?	.	?	?	?	?	?	.	.	?	P	D	D	D	D
Dev't Env'm't	.	Linker(ld)	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	Lib'n (ar)	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	ldutils	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	msgval	?	?	?	?	.	?	?	?	?	?	.	.	?	T	D	P	P	D
Diagnos	.	Monitor	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
Test	.	Sys Excisr	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
OZIX Exec.	.	Boot/Recov	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	Debug Svr	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	V-Nub	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	Process	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	Mem. Mgmt	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	.	OSF API	?	?	?	?	.	?	?	?	?	?	.	.	?	T	D	P	P	D

OZIX component translation decisions table (cont'd)

Class	IE	Component	Language Group: UK	GY	FR	SP	IT	SW	NO	DK	FN	PO	NL	IS	AR	JA	SC	TC	KR	TH
OZIX	.	FIFOsubsys	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
Exec Cont'd	D	Security	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
File System	.	Name Systm	?	?	?	?	.	?	?	?	?	?	.	.	?	T	D	P	P	D
	D	Log/Recvr	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
Netwrkg	.	Internet	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
	.	Skt & XTI	?	?	?	?	.	?	?	?	?	?	.	.	?	.	.	.	.	.
	.	NFS	?	?	?	?	.	?	?	?	?	?	.	.	?	T	D	P	P	D
	D	Kerberos	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
Terms, I/O	.	Term Suppt	?	?	?	?	.	?	?	?	?	?	.	.	?	T	D	P	P	D
	.	LAT	?	?	?	?	.	?	?	?	?	?	.	.	?	.	.	.	.	.
	.	I/O Config	?	?	?	?	.	?	?	?	?	?	.	.	?	.	.	.	.	.
	.	PIPE	?	?	?	?	.	?	?	?	?	?	.	.	?	.	.	.	.	.
	.	CSMA/CD	?	?	?	?	.	?	?	?	?	?	.	.	?	.	.	.	.	.
Mass	.	Storage	?	?	?	?	.	?	?	?	?	?	.	.	?	.	.	.	.	.
RPC	.	RPC	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
Sys/Net Mgmt	D	Sys Admin	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	D	Sys Instln	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
Perfmce	.	Monitor	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
	D	Act. Reptg	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
	D	PROF/GPROF	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
	.	Instrumntn	?	?	?	?	.	?	?	?	?	?	.	.	?	?	D	D	D	D
Layered Prods.	D	LMF	?	?	?	?	.	?	?	?	?	?	.	.	?	D	D	D	D	D
	T	DECwindows	?	?	?	?	D	?	?	?	?	?	T	T	?	T	D	P	P	D
	.	CASE tools	?	?	?	?	.	?	?	?	?	?	.	.	?	P	D	D	D	D
	.	Prog.Tools	?	?	?	?	.	?	?	?	?	?	.	.	?	P	D	D	D	D

3.2.1 Notes on the Translation Decision Table

Inputs from multilingual countries is shown in the table against Language Group. Special requirements are noted below.

1. Belgium: No requirement for localized versions (FR/NL)
2. South Pacific Region:No resources to give any feedback.

The following additional notes should be read in conjunction with the Asian Language groups decisions in the table.

1. For Asian languages there other language specific engineering components in to be added that are not covered under the general title "translation." The most important of these is the input methods. These are not covered adequately in the PIP descriptions, but if they were to be forcefit into the model they would go in the B components.
2. The Japanese computer market is very similar in complexity and sophistication to the US. It is therefore assumed that the entries marked D or P under Japan will become T in later versions.
3. The translation decision for Simplified Chinese can be deferred to later versions because the unrest in PRC has put their technology program on hold.
4. The translation decision for Thailand can be deferred because the computer market is immature; there is not a significant market in the areas being targeted by OZIX--commercial applications, production systems, etc.
5. There is a small but growing market for these features in Taiwan and Korea. As this market becomes more important, the entries marked D and T in these columns should be updated.

3.3 REQUIREMENTS FOR LOCALIZATION KIT

ABSS (JRD) will submit any localization kit requirements separately and later to the OZIX group.

4 SPECIFICATIONS

This section supplements the Corporate Product Specification for OZIX, and in particular forms the specification for product variants.

ABSS (JRD) will be producing specifications for Asian language versions of OZIX separately from this document.

4.1 VARIANT SPECIFICATIONS

4.1.1 Maintainability

Refer to the CSSE Servicability Requirements Document. That document should address the requirements for Worldwide support of multilingual and multinational companies.

4.1.2 Compatibility

OZIX requires compatibility at various levels with other Open Systems operating environments. Whilst multilingual-capable products have no additional compatibility requirements, multilingual capability itself is likely to make compatibility more difficult to implement.

Note: ABSS input is required on any specific J-ULTRIX compatibility requirements. It is however anticipated that where such a requirement is different from the generic requirements expressed in this plan, ABSS will need to address the differences in a market specific "C" component.

4.1.3 Evolvability

As the OZIX product evolves towards commercial, transaction-processing systems usage, it can be expected that translation requirements will increase and greater benefit will be gained from the International functionality provided in the base system.

5 PROJECT PLAN

To be supplied for any product variants required as a result of translation decisions.

5.1 PHASE PLANNER

Corporate Phase 0/1 close - 17th November 1989

Corporate Phase 2 close - to be supplied.

5.2 PROTOTYPE TRANSLATION PLAN

To be supplied if Prototype Translation required.

APPENDIX A

PROJECT DOCUMENTS

The following documents have been referenced within this Internationalisation Plan, or are included here:

- International Engineering "Producing International Products Reference Set" (also known as the PIP Handbook). A-MN-ELSM498-00-0 Rev B.
- DEC STD. 066-3 Policy for Designing Products for All Countries Designated as Strategic Markets. EL-00066-03.
- Terry Morris, OS/SB - "OZIX product and Market Requirements Document" Rev 1.0
- X/Open Portability Guide (XPGn) Versions 3 (issued) and 4 (under determination)
- ISO 10646 Multiple-octet Coded Character Set.
- IED/ABSS "Digital's Text Model for the Future - What Should It Be ?" 20th June, 1989.
- Claire Cockcroft, OS/SB - "A Vision of the International OZIX Product"
- Functional Specification for XNLS. Contact VOGON::LANDLES.
- Catherine L. Richardson, OS/SB - "Preliminary OZIX Applications Plan"
- Omur Tasar, "OS/SB Enterprise Management Approach - an Overview" 12th July, 1989.
- Internationalization Plan for DECforms V2.0 - Contact VOGON::I18N_PLANS
- Internationalization Plan for CDD+ V5.0 - Contact VOGON::I18N_PLANS
- Internationalization Plan for RdbStar V1.0 - Contact VOGON::I18N_PLANS

Colin Walters of CUP/ITG I18N has produced the following outline of a proposed documentation strategy for OZIX documentation:

```
From:     VAXUUM::WALTERS "ZK01/3j03 DTN 381-1955  04-Aug-1989 0826"

To:       VOGON::KEE,SWEEZEY,VOGON::OUGHTON
CC:       WALTERS
Subj:     Comments on the OZIX Internationalisation Plan

CC Martyn Kee, IED UI Manager
   Marcia Sweezey, CUP/ITG  I18N

   PROPOSED STRATEGY:
   ~~~~~~~~~~~~~~~~~~
   I strongly support the application of modular techniques in
   this project.  It is essential that UI engineering is involved
   early in the writing process for this to succeed.

   1.  Perform an analysis of anticipated users and write a user-profile
       for each.

   2.  Perform an analysis of OZIX commands and subsystems.  Structure the
       UI on the basis of:

       o  Which users use which commands and systems.
       o  Logical relationships between commands and subsystems.

   3.  Develop a single text-library or database which documents the
       system. This will have two-dimensional modularity:

                    ^    Subject      Subject    Subject
                    |
      MODULARITY    |    overview
      BY            |    application
      COMPLEXITY    |    reference
                    |    programming
                    |
                    +------------------------------------>
                      MODULARITY BY SUBJECT

   4.  Use conditional bookbuilds to extract text from the database and
       build into Online books, help and documentation. It will be easy
       for any Local Engineering group to extract and translate only the
       text that they require, but the integrity of the Corporate
       documentation will be maintained.

       Other advantages:

       o Customisable for the customer
       o Easier updates
       o Easier support
       o Eliminate redundancy
```

At the end of August four companies doing business in Japan sent representatives to four days of ad-hoc meetings to flush out their i18n requirements for OSF/1 OSC.

Participants represented YHP, IBM-J, Hitachi, and DEC-J. OSF/PO folks attended the introduction and wrap-up sessions. The requirements list that resulted from these meetings will be reviewed at the OSF I18n SIG meetingto be held in Cambridge, September 22-23. It is inserted here in support of the OZIX Internationalization Plan.

Internationalization/Localization Requirements to OSF/1 OSC

1989-08-24
Internationalization Ad Hoc Group in OSF/PO

This requirements list has been elaborated by the Internationalization Ad-hoc Group composed of DEC Japan, Hitachi, IBM Japan and YHP. This is concentrated on OSF/1 OSC. This list is intended to be a contribution to OSF Internationalization SIG. The meaning of the priority levels are as follows.

Priority Level 1 : To be included in OSC release 1.0 (Minimum Requirements)

- Based on XPG3,

  Extended to be able to handle multibyte characters. Functions included in this level are expected to have no deviations from the anticipated XPG4 functions.

Priority Level 2 : To be included in OSC release 1.1

- XPG4 minus Minimum Requirements

Priority Level 3 : To be included in the subsequent releases of OSC release 1.1

- Beyond Level 2 functions.

Internationalization

[General]

1. OSF/1 Internationlization specification should be independent from coded character-sets. (pr = 1)
2. Single source-code for all of OSF/1. #ifdef should not be used in Internationalization functions. (pr = 1)
3. Conformance to Internationalization functions specified in X/Open XPG3, ISO DIS 9945 (POSIX 1003.1 Draft 8). (pr = 1)
4. OSF/1 should handle a multibyte character as a character, not as a byte. (pr = 1)
5. ERA/emperor year should be supported. The date/time format should support multibyte characters. (pr = 2)

[Kernel]

1. Erase operation in tty module should be performed on a character basis instead of on a byte basis. (pr = 1)

[Command and Utility]

1. Multibyte characters should be supported in filenames and directory names. (pr = 1)
   (note) Characters which have special meaning in directory name may be limited to ASCII.
2. All commands supported by OSC release 1.0 should handle multibyte-character filenames and messages. The commands/libraries listed in Table 1 should support the local environment. (pr = 1)
3. Regular expression pattern-matching should be performed on the basis of characters . (pr = 1)
   (note) Meta-characters may be limited to ASCII for the time being. Range expression and character class are to be defined.

[Language and Library]

1. C compiler must satisfy and include the multi-byte handling functions specified in ANSI C (X3J11). (pr = 1)

Localization for Japan

1. Japanese characters should be supported in commands/libraries listed in Internationalization 8. (pr = 1)
2. There should be support for coded character-sets currently used in Japan, for example; EUC/UJIS and Shift JIS. (pr = 1)
3. OSF needs to have code-conversion libraries/filters, for example; EUC/UJIS to/from Shift-JIS. (pr = 2)
4. Formatting for writing direction (vertical writing). (pr = 3)

Others

1. Support for the Asian languages. (pr =)
2. Translation of messages and on-line manuals into the local language. (pr =)

adb	diff3	nroff
admin	dirname	od
at	echo	page
awk	ed	pg
basename	edit	pr
batch	egrep	ps
bdiff	ex	red
calendar	expr	rm
cat	fgrep	rmdir
cb	find	sdb
cc	fold	sed
cdc	gencat	sh
cflow	get	sort
col	grep	tabs
comm	join	tar
cp	ksh	tbl
cpio	lint	tee
cpp	ln	test
csh	ls	tr
cu	mailx	tsort
cxref	make	uniq
date	man	vi
delta	more	view
deroff	mv	wc
diff	news	what

Table 1

APPENDIX B

OUTSTANDING ISSUES

A minimum subset of fonts required to ship worldwide with the base product needs to be identified in conjunction with the group producing the Font Access Facility (FAF).

October 1989

OZIX Version 1.0
System Product Description

Revision Information:

Version 1.0

Written By:

Kathy Appellof, *OZIX Product Management*

Contributors:

Terry Morris, *OZIX Product Management*
Members of the OZIX Project Team

Digital Equipment Corporation
Confidential and Proprietary

TABLE OF CONTENTS

Preface

This document is a description of the OZIX Version 1.0 Product. It is intended to be a high-level overview.

Change History

Date	Issue #	Description
October 1989	0.1	OZIX System Version 1 Product Description
October 1989	0.2	OZIX System Version 1 Product Description
October 1989	1.0	OZIX System Version 1 Product Description

Date of Printing:

27-OCT-1989 10:36:25.66

OZIX SYSTEM DESCRIPTION

OZIX is an extensible, hardware-independent operating system that integrates modern technologies for distributed systems, fault tolerance and data integrity, while adhering to a high-performance I/O implementation. Its capabilities will evolve over a series of releases and are composed of system elements from other Digital groups, (such as database and transaction processing software, compilers, tools, and utilities) as well as selected elements from third-party vendors. OZIX is a reliable, high-performance production system that is targeted at the open systems market. While transaction processing technologies are built into Version 1, these capabilities will not be realized until a release after Version 1. In the first release of this product the system utilizes these transaction technologies in demonstrating its high level of performance and reliability.

OZIX functions as a network server in a multivendor, distributed systems network. It serves this network by providing:

- File services via NFS
- Compute services for user applications
- Workstation services such as workstation booting and diskless services.
- Database services

The OZIX system is designed with the methodology required to obtain a B2 level security rating from the National Computer Security Center (NCSC), thus ensuring strong resistance to intrusion and the protection of data from unwanted disclosure.

The OZIX product also provides an internationalized computing environment capable of supporting applications that span national, linguistic, and cultural boundaries. This support is apparent throughout the system - in libraries, terminal services, the file system, the base system, and the message facility.

OZIX is a modern operating system platform that integrates existing and emerging open system standards defined by POSIX, X/Open, OSF and ISO/OSI. Support for standards - such as those defining user interface, data exchange, networking, file access, multi-processing, transaction processing, security, system administration, and internationalization - are built into every component of the OZIX system.

Also included in OZIX Version 1 are integrated system and network management features, license management, and CDROM distribution.

OZIX application interfaces provide a development platform for production applications. Portability of applications within a Digital environment is provided with Application Integration Architecture (AIA) components such as DECwindows client libraries, Compound Document Architecture (CDA), and Concert Multithread Architecture (CMA).

OZIX USER ENVIRONMENT

Users access OZIX through an interactive interface, by logging into the system using either character-cell terminals or bitmapped terminals and workstations. The type of interactive interface the user sees is determined by the method used to access OZIX. The OZIX command line interface is characterized by use of a command interpreter called a shell. This shell presents a programmable interface to all OSF-specified standard commands and utilities, plus additional Digital-standard and OZIX-specific commands and utilities.

The OZIX bitmapped terminal interface is provided to users through the DECwindows client interface. DECwindows support is a combination of "X", Motif, and Digital value-added. Client access is executed by OZIX connections to ULTRIX, UNIX[1], and VMS.

1 Commands and Utilities

A number of commands and utilities for general users are provided with the system. These commands and utilities comply with the following industry and de facto standards:

- X/Open Portability Guide, XSI Commands and Utilities (Issue 3)
- POSIX 1003.2, Shell and Application Utility Interface for Computer Operating System Environments (draft 8)
- OSF Operating System Component (OSC) Functional Outline: Application Environment Specification (Revision 1.0)

End-users will most commonly use these commands and utilities to perform such tasks as maintaining files and directories, communicating with other users and systems, processing of documents, printing of documents, and obtaining system information.

2 OZIX Shells

There are two shells provided with the OZIX system - the Bourne Shell and the C Shell. These shells provide a line-oriented command interface and functionality that is identical to other implementations provided on UNIX, ULTRIX, and OSF systems. Users may select a default shell or change shells dynamically. The shell preference is stored as part of the per-user data managed by the system administration components.

[1] UNIX is a registered trademark of American Telephone and Telegraph

OZIX Program Development Environment

3 OZIX Program Development Tools

The OZIX CASE environment is based on the use of standard programming tool X/Open, OSF, and POSIX.

Application developers can use all of the commands and utilities used by end user commands and utilities to perform tasks on OZIX. These tasks include:

- Creation and compilation of source code
- Link and debug of applications
- Application performance analysis
- Creation of utility programs and sophisticated shell scripts
- Distribution of applications using remote procedure call technology

4 The OZIX C Compiler

The features of the OZIX C compiler include:

- Compliance with X3J11 ANSI C definition
- Conformance with X/Open, and POSIX standards
- Support for traditional C and VAX C features
- Support for internationalization capabilities
- Support of mixed language interoperability
- Code portability with other existing Digital C implementations

This compiler employs advanced local and global optimizations including scalar opt procedural analysis, procedural inlining, pipeline scheduling and register allocatior

5 Application Programming Interfaces

5.1 Standards-compliant Application Programming Interfaces

OZIX provides a set of application programming interfaces that are compliant with standards:

- X/Open XPG3
- OSF AES
- POSIX 1003.1
- ANSI C

Binary code for these APIs is provided with the system in two forms: as objec files and also as shared libraries. Users normally prefer to link with shared libra standard library routines provided with OZIX include:

- Standard ANSI C
- Language runtime support
- Math
- Standard UNIX routines

- Primitive DBMS routines
- GKS and PHIGS
- X11 toolkit and widget support
- OSF/Motif support

5.2 Digital Added-value Application Programming Interfaces

Also included is a set of Digital added value APIs, including:

- DECwindows Toolkit routines
- Compound Document Architecture (CDA) DDIF (Digital Data Interchange Format) routines
- Compound Document Architecture DDIF Viewer routines
- Concert Multithread Architecture routines
- DECprint interface
- Application Portability Architecture routines
- DECnet programming routines
- Compound string support
- Online Diagnostic Monitor (ODM) interface routines
- OZIX extensions to messaging
- OZIX Security Support Component routines
- Logging and error recovery system routines

OZIX System Administration

6 System Administration

The OZIX system administration design addresses the requirements of both production systems and management of B2 security policies. The OZIX system administration structure is designed into each component of the system in an integrated fashion.

The OZIX system administration structure is based on an object-oriented methodology in conformance with Digital's Enterprise Management Architecture (EMA). All tasks performed by system administration are performed on manageable objects, that represent users, devices, processes, networks, and so on. As objects, these components are identified by their specific attributes, operations, and events.

A management backplane is used to interconnect manageable objects to each other and to management applications. The same EMA-compliant management backplane is used by both OZIX system administration and network administration providing a common set of services across interconnected systems.

6.1 Management Applications

Management applications request management operations to be performed. The request for an operation is made by the management application, through the management backplane, to the object, Objects interact indirectly with management applications and have no knowledge of the management applications.

The types of management applications are unlimited. Some management applications are simple user interfaces using command line and/or DECwindows technology. Other management applications may be more complex decision making applications that issue management operations based on a set of rules. As for the object being requested to perform some operation, the type of management application is immaterial.

OZIX provides a set of user presentations and a limited number of decision making applications for resource and fault management.

6.1.1 User Presentations

A user presentation is a management application providing an interface that system and network managers use to manipulate manageable objects. A set of user presentations are implemented to provide a variety of interfaces for use through character-cell terminals, shell command scripts and DECwindows terminals.

6.1.1.1 DECwindow Storage Manager

The DECwindow Storage Manager allows monitoring and control of manageable objects in OZIX Storage Management that is based on the Attribute Based Allocation (ABA).

6.1.1.2 Management Command Language (MCL)

MCL is a user presentation management application providing a command line style interface and a generic DECwindows style interface to all management objects on OZIX. A subset of MCL satisfies B2 security requirements of a trusted path for the security administrator.

6.1.1.3 Standards-Compliant Commands

Some OZIX system administration functions are available through a standards-compliant interface.

OPERATING SYSTEM ENVIRONMENT

The OZIX base system is designed to deliver maximum performance, reliability, and security. By incorporating state-of-the-art technologies that deliver superior availability and integrity of data across a large number of storage devices, the base system provides a solid foundation for transaction processing and customized applications. It consists of an executive and a small hardware-dependent portion of code that is called the *nub*.

7 General Executive Features

Most of the base system support that is in the operating system is provided by the executive. This includes process management, memory management, and the file system. Internal features that support these functions are:

- Fine-grained locking improves concurrency in multiprocessor environments.
- Multithreading facilitates high performance.
- Isolation between subsystems provides security and integrity.
- Level checking in locking assures deadlock prevention.
- Preemptable executive allows for fast response.
- Scalability of system resource limits
- Pageable executive code and data provides better physical memory utilization.
- Flexible and extensible memory allocation and management.
- Stack-based exception handling in the operating system for increased reliability.
- OZIX is designed for multiprocessor systems with large amounts of memory, I/O devices, and users.
- Integrated transaction processing technologies

7.1 Process and Process Environment

OZIX provides a number of standard UNIX features in the executive. This includes standard UNIX process management system services, UNIX signal interfaces, and UNIX environment interfaces. Concert Multithread Architecture core services, shared libraries, scheduling features, and interprocess communication (IPC) calls are Digital added-value features that are also included in the OZIX executive.

7.2 Memory Management

The OZIX executive supports the OSF API memory management interface. Mapped files and data in the file system are automatically kept constant, which means that if an application opens with read/write access while another application owns the mapped file then they will recognize each others changes. Other memory management features include:

- Unlimited number of virtual address spaces, subject to amount of physical memory
- Efficient management of sparse virtual address space
- Multiple page files
- Maximize size of user virtual address space
- Alternative backing store managers and page replacement algorithms

- Maximize use of physical memory
- Physical memory sharing

7.3 Transaction Processing Services

The OZIX executive provides services which support transaction processing. Executive components, such as the file system, use these transaction processing services.

7.4 Multithreading

OZIX provides multithreading capabilities as specified by the CMA. This architecture defines portable services for creating and controlling multiple threads of execution within the address space provided by a single process.

Multithreading brings numerous advantages to application programs. On a system which provides hardware and software support for parallel execution, the use of threads can speed up execution of an application program by providing it with the ability to utilize all the processors simultaneously.

7.5 Symmetric Multiprocessing

OZIX provides symmetric multiprocessing (SMP) support, which is a form of tightly coupled multiprocessing in which all processors perform operations simultaneously. SMP configurations consist of multiple CPUs executing code from a single shared physical memory. Users and processes share a single copy of the OZIX system. Operating system code can be executed on any processor. Different threads accessing the same data structures, can be executing simultaneously on multiple processors. This is accomplished safely by means of locks, which are used to control the concurrent access of shared data structures.

8 Input/Output System

OZIX mass-storage and terminal subsystems are designed to meet the requirements of transaction processing systems, which process hundreds of user transactions per second. These systems often process large volumes of transactions which require a large number of mass-storage devices and terminal sessions.

8.1 Mass Storage

The OZIX mass storage structure provides the basis for high I/O throughput by minimizing latency. It supports DSA (both DSA-1 and DSA-2) devices. DSA storage devices are accessible via the Computer Interconnect (CI). Different types of mass storage devices are supported, including all DSA devices and CDROMs.

The OZIX I/O resource manager provides the mechanisms to manage the hardware and software components of the OZIX mass storage system. The management interface to the I/O resource manager provides capabilities that:

- Automatically configure physical devices at system startup
- Configure/autoconfigure new physical devices on a running system
- Cooperate with other system software in providing warm restart capabilities

8.2 Storage Model

OZIX is designed to efficiently manage tens of terabytes of data on hundreds of storage devices. The task of managing such massive storage resources is made possible by integrating Digitals Attribute-Based Allocation (ABA) architecture into the design of the I/O and file subsystems. This architecture allows data to be dynamically mapped to storage devices. This dynamic allocation of data provides the foundation for features, such as advanced storage management and dynamic reconfiguration of storage devices.

8.3 File System

The OZIX file system provides reliable access to files and data through the POSIX 1003.1 file system interface. This POSIX interface is extended by the addition of resource management support services, which allow support of database systems in a transaction processing environment. On disk data structure updating is optimized through the use of buffer management and logging techniques. The underlying file system is based on the attribute-based allocation (ABA) architecture, which provides support for a large number of disks and advanced storage management techniques. OZIX provides better utilization of storage devices through migration of files between devices in a hierarchy of devices. The OZIX file system also provides mandatory and discretionary access controls for files.

8.4 Terminals

The OZIX terminal subsystem provides a means for configuring and operating a very large number of terminal sessions. It utilizes the Multioctet Character Set (MOCS) in the support of many of the worlds languages. OZIX provides inherent support for multiple kinds of input methods with at least U.S. and Japanese Kanji input for V1.

The terminal subsystem consists of a terminal class driver and a terminal port driver. The terminal class driver provides user sessions as well as functions for data presentation. The class driver follows presentation rules as specified by POSIX standard 1003.1. The terminal port driver moves uninterpreted data between a hardware interface or network software module and the terminal class driver.

Terminals can be connected to an OZIX system in a number of ways. OZIX supplies drivers to support the connection of the system console terminal, Local Area Transport (LAT) terminals, and network terminals. For network terminals, OZIX supports the TELNET, rlogin, and CTERM protocols.

9 Distributed Processing

OZIX provides native implementations of TCP/IP and of DECnet/OSI. TCP/IP provides Internet access for interoperability with ULTRIX and other UNIX systems. DECnet/OSI provides interoperability with VMS and ULTRIX as well as other vendor OSI implementations. The interface to both networks includes BSD sockets and the X/Open Transport Interface.

A native implementation of NFS (Network File System) based on the Sun V2.0 protocol is provided on OZIX for transparent file access across ULTRIX, OZIX, VMS and other UNIX vendor systems. This implementation provides server failover in support of OZIX availability goals.

9.1 Services

Services available with the OZIX system include:

- Authentication support for clients and servers by use of Kerberos and DASS
- Distributed name services are provided through DNS (distributed name services) and BIND. An integrated DNS/BIND/Hesiod name server is a preferred configuration.

- X.400 mail services
- Distributed time services via DTSS and NTP

9.2 Remote Procedure Calls - DECrpc

OZIX provides DECrpc as a remote procedure call (RPC) mechanism for developing distributed applications. DECrpc consists of three major components:

- Stub Generator: Compiles interface definitions and produces client stubs and server stubs. These stubs are then compiled with the service consumer and service provider parts of the application, respectively. Much of the communication and data packaging is handled within the code produced by the stub generator, and therefore hidden from the user.
- Name Server: DECrpc utilizes the location broker for naming services. These naming services manage a database of servers and interfaces so that clients requesting a particular service can locate a server with which to bind.
- RPC Runtime: These routines are linked with the client and server applications. They provide services to look up interfaces, establish and break client/server bindings, and register/deregister interfaces. Some of these routines are included in the code produced by the stub generator and are never actually referenced by the user-written part of the applications.

9.3 Network Management

The components of OZIX network management are designed to provide support for local and remote management of both DECnet/OSI and TCP/IP network components. Supported components of OZIX network management are:

- The *Maintenance Operations Protocol* (MOP) is used to downline load software to Digital systems.
- The *Network Management Director* is the user interface to network management.
- The *Event Dispatcher* is the component of network management that receives notification of state changes from the various network entities and routes them to logging and reporting systems.
- The *Entity Interface* portion of network management provides network entities with a standardized interface to network management.
- The *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) implements the entity management functions of the CSMA/CD service in a device-independent manner.

10 Resource Management

The OZIX design provides a resource and fault management strategy for a wide range of error detection, correction, and isolation. This incorporates rule-based failure prediction, error detection, error logging, fault analysis, and system configuration control to predict, detect, and when possible correct hardware and software errors.

OZIX features that enhance system reliability include a data logging mechanism, disk shadowing, and the ability to dynamically reconfigure storage devices.

11 Performance and Instrumentation

The OZIX system is heavily instrumented and provides a rich set of instrumentation services. In addition, OZIX has a data collection facility, a low-level software component, that enables the system to collect component and system-wide data. The Instrumentation Data Collection Facility (IDCF) collects performance and utilization data maintained by OZIX. This facility supports two data collection methodologies - event detection and sampling.

- Event detection is used when it is necessary to know a sequence of events or the exact number of their occurrence over a given interval of time. OZIX employs software probes for event detection, which are used for the recording of an event.
- Event sampling is used to observe rates and frequencies of events.

OZIX provides performance characterization tools which enables OZIX-based systems and their constituent components to be characterized with a high degree of accuracy. Traditional utilities and commands are also provided.

12 Security

OZIX is designed with National Computer Security Center (NCSC) B2 methodology. It provides high levels of system security and integrity. OZIX supports multiple security and integrity policies to satisfy the needs of both government and commercial environments. Each of these security and integrity policies are enforced with the full strengths of a B2 system.

Security supported provided by OZIX gives system administrators flexibility in tailoring the security profile of each individual OZIX system. Local security policy can be implemented using the following types of controls:

- Access Control Lists (ACLs)
- Mandatory Access Control (MAC)
- Integrity Access Control (IAC)
- Level control of network access
- Control over access to "raw" devices

In addition, OZIX security support allows the security administrator to change the basic security model in effect on the system. For example, some OZIX systems may use a security model based on the Bell and LaPadula model, or may allow the use of a customized security model.

13 Internationalization

OZIX supports characters for most every language through the implementation of MOCS. In MOCS, up to four octets (8-bits) can be used to define a single character for languages such as Japanese, Chinese, and Korean.

OZIX commands use multilingual messaging and language switching facilities, thus accommodating easy translation by local engineering groups. Selected OZIX commands are fully internationalized and have the capacity to process data and file storage objects presented in character sets other than the ISO 8859-1 character set (8-bit Latin-1).

OZIX supplies a set of standard C libraries to provide string handling routines to deal with string parameters made up of 8-bit characters. Existing C applications can link against the OZIX standard C libraries for compatibility. OZIX extends the standard C library to provide support for compound string technology, allowing development of new international applications that are language neutral.

The OZIX terminal subsystem provides support for multiple character sets and languages. The terminal subsystem allows for the inclusion of additional locale-specific character set handling without modification of the basic subsystem code or data structures. Existing applications may access the terminal subsystem through the OZIX OSF API.

14 Standards

The basic system concepts of OZIX reflect the models of process management, file processing, and system calls defined by X/Open, OSF, POSIX, OSI, and ANSI standards.

15 Documentation

The OZIX information set is structured specifically for online presentation in a hypertext information style. Primary documentation retrieval method for OZIX is online. Hardcopy documentation is also available.

The OZIX information set is tailorable for the needs of different types of users and is designed for easy translation to other languages. Publication tools are provided to allow for integration of training modules and documentation from other sources such as third-party vendors.

October 1989

OZIX Version 2.0 Working Paper

Document Version:

Version 1.0

Written By:

Kathy Appellof, *OZIX* Version 2.0 Product Manager

Contributors:

Garth Reid, *DBS Product Management*
Gary Shroyer, *TP-WEST Operations*

Digital Equipment Corporation
Confidential and Proprietary

TABLE OF CONTENTS

FIGURES

TABLES

Preface

The OZIX Version 2.0 Working Paper contains information discussed and agreed to by members of the Open Production Environment SIG. Organizations represented on this SIG group include OZIX Product Management, Database Systems Product Management, System Software Marketing, and TP-West Engineering.

This document describes the strategy for the OZIX Version 2.0 program. This program includes the OZIX operating system, a relational database (RdbSTAR), and transaction processing (DECxtp) functionality. Together, these are the foundation for Digital's entry into the Open Production Systems Transaction Processing market.

Associated Documents

1. OZIX Vision Document
2. OZIX Market and Product Requirements Document
3. OZIX Applications Requirements Document
4. TP Market Sizing Document

Change History

Date	Issue #	Description
July 1989	0.1	Preliminary OZIX V2.0 Strategy Document
August 1989	0.2	Preliminary OZIX V2.0 Strategy Document
September 1989	0.3	OZIX V2.0 Working Paper
October 1989	1.0	OZIX V2.0 Working Paper - incorporate changes from primary review and editing

OZIX SYSTEMS PROJECT TEAM

OZIX Systems Manager:

John Gilbert, *OZIX Engineering Manager*

OZIX Engineering Team:

Benn Schreiber, *OZIX Project Manager*
Marilyn Fries, *OZIX Development Supervisor*
Dick Funk, *OZIX Development Supervisor*
Mark Ozur, *OZIX Development Supervisor*
John Penney, *OZIX Development Supervisor*
Mike Peterson, *OZIX Development Supervisor*
Dave Snow, *OZIX Development Supervisor*
Bill Watson, *OZIX Development Supervisor*
Lu Anne Van de Pas, *OSG Compiler Development Supervisor*
Steve Jenness, *OZIX Network Architect*
Chris Saether, *OZIX ABA Architect*
Jim Schirmer, *OZIX Security Architect*
Claire Cockcroft, *OZIX Internationalization Program Manager*
Pete Benoit, *OZIX Project Leader*
Richard Brown, *OZIX Project Leader*
Sumanta Chatterjee, *OZIX Project Leader*
Jan D'Addamio, *OZIX Project Leader*
Mark Ditto, *OZIX Project Leader*
Dennis Doherty, *OZIX Project Leader*
Min-Chih Lu Earl, *OZIX Project Leader*
Debbie Girdler, *OZIX Project Leader*
Kelly Green, *OZIX Project Leader*
Jeff Havens, *OZIX Project Leader*
Brett Helsel, *OZIX Project Leader*
Oscar Newkerk, *OZIX Project Leader*
Charles Olivier, *OZIX Project Leader*
Jim Teague, *OZIX Project Leader*
Dave Walp, *OZIX Project Leader*
Charlie Wickham, *OZIX Project Leader*

OZIX Finance:

Judy Fox, *OZIX Business Support Manager*
Maggie Schimpf, *OZIX Financial Manager*
Salley Anderson, *OZIX Financial Analyst*

Product Management:

Mike Parker, *Manager OZIX Product Managers*
Kathy Appellof, *OZIX Transaction Services Product Manager*
Terry Morris, *OZIX Product Manager*
Cathie Richardson, *OSG C/C++ & Applications Product Manager*

Manufacturing:

Rose Ramsey, *Software Manufacturing Product Manager*

Documentation:

Jim Jackson, *OZIX Documentation Project Leader*
Liz Hunt, *OZIX Application Programming Environment*
Marcia Aguero, *OZIX System Management Environment Project Leader*
Bill Talcott, *OZIX Software Support Environment Project Leader*
Cheryl Snyder, *OZIX Usability Engineering/Testing Project Leader*

Customer Services:

Bill Hilton, *OZIX Customer Services Systems Engineering Manager*
Thomas Siebold, *OZIX CSSE Maintainability Engineer*
LeeAnn Stivers, *OZIX CSSE Product Manager*
Myrna Harrison, *OZIX ESDP Project Leader*

Date of Printing:
27-OCT-1989 11:26:52.76

1 Introduction

The UNIX ™ operating system is being used extensively as the technology base for recent system introductions into the production systems market, encompassing transaction processing (TP), batch, and timesharing systems. This trend is expected to continue well into the next decade as major vendors rush to gain the competitive edge in this market.

Vendors considered to be active contenders in this market invest heavily in the addition of features to the UNIX operating system in overcoming the weaknesses of the base operating system.

This document explores the reasons why the UNIX system is gaining the attention of the commercial market, the evolvement of this into the open systems market, and an overview of its major strengths and weaknesses.

The OZIX product family strategy is then discussed as Digital's entry in this market.

1.1 UNIX System Summary

The UNIX operating system first appeared around 1970 at Bell Labs. UNIX was primarily used by AT&T as a base for text processing applications and remained an internal product at Bell Labs until the mid-1970s. The UNIX system was written in macro and the C language. It was eventually released worldwide to the academic community, thus facilitating its emergence as a popular operating system in the business community.

For AT&T, the UNIX system provided the extensibility required by their in-house production systems. These systems were not considered to be transaction processing oriented, as transaction processing requirements of production systems were not a major concern.

Independent software manufacturers quickly realized that they could take the best of the UNIX system and easily develop variations of it for their new systems, thus greatly reducing development time. This reduction in development time is believed to have been an influence in the introduction of new technologies such as RISC (Reduced Instruction Set Computing).

1.2 Open Systems Summary

An open systems environment is a comprehensive, consistent set of international technology standards and profiles, specifying interfaces and supporting formats for interoperability or portability of applications and people.

Market demand for open systems is heavily influenced by many open software organizations such as X/Open, OSF, POSIX, and OSI. One of these organizations, the Open Software Foundation (OSF), was founded in May 1988 to develop, license, and distribute an open software environment that is based on an independent UNIX-derived operating system and related subsystems. OSF addresses three major issues in the open systems market: portability, interoperability, and scalability. Applications written to the OSF standards are portable from one architecture to another. Interoperability provides the ability to exchange data and run applications on computers from multiple vendors. Scalability provides the means to migrate software across different hardware classes. By specifying an environment built on standards that provides these capabilities, the OSF fulfills customers' demands to protect their long-term software investment.

™ UNIX is a registered trademark of American Telephone and Telegraph

1.3 Online Transaction Processing Summary

Traditionally, transaction processing has been regarded as a method of computing for large, corporate IBM mainframe systems. Some of the first online transaction processing systems designed were the airline reservations systems. These systems tended to be highly complex in nature requiring large mainframe systems. They are are now approaching twenty-five years in age, yet are still considered to be highly effective. Other traditional transaction processing examples include telecommunications (area code dialing, 911), automatic teller machines, and bank debit cards.

Major companies also realized the major cost savings realized by integrating OLTP features into corporate databases. Applications such as inventory management in the Computer Integrated Manufacturing (CIM) market proved that transaction processing systems could completely manage order processing and reduce inventory, resulting in greater company profits.

Applications such as the above have lent themselves to the definition of certain characteristics of TP systems. TP demands high system availability, the ability to process large volumes of data against large databases, strong data integrity, and most importantly, fast I/O throughput. TP applications require optimized disk, memory, process, and data management to meet these demands.

Computer manufacturers quickly realized the competitive advantage gained by the merger of transaction processing functionality with UNIX-based systems. Tandem and Stratus are excellent examples of companies who currently own the competitive advantage in the fault-tolerant transaction processing arena; other active contenders include SUN, Hewlett-Packard, Pyramid, and NCR.

TP systems have emerged during the late-1980s as strategic computing platforms. On a worldwide revenue basis, the TP market is estimated to be growing at a compound annual growth rate of 13.5 percent [1]. Along with this growth, UNIX systems are also emerging as "standard" operating systems. This is evident in recent hardware/software vendor announcements linking these two technologies.

2 Why Database Vendors are Drawn to the UNIX System

Even though it has identifiable weaknesses in the open systems market, there are reasons why database vendors turn to the UNIX system in this non-traditional UNIX market.

2.1 Vendor Profile

Primary attributes of the UNIX system that draw third-party DBMS vendors and end-users include:

- Portability. Application software written for the UNIX system on one computer is easily ported to other computers that also run UNIX. This results in lower development cost to the vendor when providing application software across multiple platforms.
- Open systems. UNIX has become an industry standard for minicomputers, multiuser microcomputers, and workstations. Information on how it works and where it is headed is widely available. This protects developers and users from being tied to a single vendor's hardware product.
- The accessibility of developer tools. As UNIX evolves, more programming aids are added to it.
- Ease of use for program development. In UNIX, devices are treated like files being written to or read from. Input/output are stream oriented. Pipes, filters, and redirection enable programmers to build large programs from smaller ones.

[1] Source: TP Market Sizing Task Force, September 1989

3 UNIX System Weaknesses in the Transaction Processing Market

Even though UNIX systems are highly desired as operating system platforms, they are still used as a basis by many competitors for their TP offerings in spite of its weaknesses. Most competitors have taken advantage of the ease of customization of the UNIX operating system to bypass the problem areas. Areas of weakness include file/disk management, process management/scheduling, buffer I/O management, data/records management, and lack of applications tuning. [1]

3.1 File/Disk Management

The TP market requires that applications have the ability to work with a small number of very large files. UNIX system optimization handles large numbers of very small files. Because of the methods the UNIX system uses for storing files in non-contiguous areas, portions of files are located widely across a disk, causing poor file access performance. It would be more advantageous for parts of data files to be stored contiguously, or at least close to each other since database file access is often sequential.

Oracle, Sybase, Informix, and AT&T give developers the option of using "raw devices", bypassing the regular UNIX system file management facilities, to directly write to or read from the secondary storage device. Jerry Baker of Oracle explains that "Typically you can get anywhere from 15 to 20% performance improvement by writing directly to disk and letting Oracle manage the data blocks instead of letting the UNIX system do it".

3.2 Process Management/Scheduling

The UNIX system is not well suited for TP Process Management. By design, the UNIX process scheduler follows a "round robin with multilevel feedback" algorithm. The UNIX time sharing algorithm generates many machine instructions per transaction, thus resulting in lower performance.

To optimize performance, an operating system should strive to minimize the number of context switches, take advice from the application regarding preemption and process scheduling, minimize the number of processes competing for CPU time, and avoid locking up system resources while processes are swapped in and out of the CPU. Database updates and queries should, ideally, complete all of their transaction processes before being swapped out, regardless of time quantum limits. To accomplish this, DBMS vendors must provide their own means of task scheduling to replace or complement UNIX's process scheduler.

3.3 Buffer I/O Management

The UNIX system is weak in buffer management. The provided algorithm for cache management is inappropriate for database applications. The UNIX prefetch mechanism detects when a file is accessed sequentially and brings extra blocks of data from that file into memory with each I/O. However, prefetch is better done at the database level. The database should be smart enough to understand what data is needed, before it is needed, and what data can be removed from memory.

3.4 Data/record Management

The UNIX system does not support record management as other operating systems do. It is limited to some file management functions. All record retrieval, locking, and file management functions must be handled by the database. Most vendors understand this and offer data management through their databases.

---

[1] "Shaping UNIX for Transaction Processing", Ron Carnahan, *Database Programming & Design*, March 1989

3.5 Tuning of Applications

The UNIX system does not provide tuning capabilities for TP applications. To optimize performance on specific applications, database vendors have been able to provide some degree of end-user tuning. Self-tuning mechanisms for maintaining good operating system and application performance are essential in the UNIX TP market. Application performance analysis, optimization, and characterization tools are also important.

4 Essential Criteria for Transaction Processing Systems

The Gartner Group has defined 10 essential requirements that system vendors must fulfill to become a major player in this market. These are used by Gartner as a measurement of an effective TP system. They expect that no single vendor will be superior in all of these categories. Vendors who are strong in most of these areas are expected to be far superior in this market. These requirements are not listed in any specific priority order.

- Availability. Users need to understand the economic impact of downtime and resulting losses should be quantified. This segment will be demanding highly-available systems, that may also tend to be fault-tolerant systems.
- Scalability. Systems must be expandable over wide performance ranges without requiring the swap out of system boxes. Microprocessor-based mid-range systems will supplant older minicomputer architectures because of the scalability of microprocessor systems.
- Distributability. Vendors must be able to offer distributed, end-user oriented systems closer to the point of contact with customers. These systems may have links to the corporate databases.
- Integrity. When systems are distributed, vendors must be able to provide facilities that ensure that transactions are not lost and are updated in near real time without causing inconsistencies in related databases.
- Recoverability. When system errors or failures occur, the system must be able to recover without loss of data or work in progress.
- Security. The system must ensure the security of corporate data, both at the desktop and at the corporate level.
- Flexibility. Systems must be reconfigurable according to evolving corporate needs. Network topologies must be independent of, and allow for such reconfigurations.
- Productivity. New desktop TP systems must offer improved application development tools such as 4GLs and CASE, that will allow applications to be distributed to both the desktop and the server system.
- Operability. System/network management must be extended to the desktop and must allow even unskilled personnel to monitor and administer networks of workstations.
- Connectivity. Systems must be integrated into both the corporate backbone enterprise network and the user sub-networks.

5 Other Significant Market Trends

As the UNIX market is evolving towards the open production systems market, it is equally important to examine other trends being seen in the market.

5.1 Open Systems

Many of the applications in use today run on old-technology systems that are expensive to use and maintain. Customers are hesitant to move these applications to newer technology because they fear the difficulties and costs of conversion, retraining, and parallel operation. However, customers see open systems, both hardware and software, as a way to use the new technology and to pay the cost of conversion once.

Open systems comply with comprehensive suites of formal standards, providing high levels of interoperability and portability.

Digital's System Software Marketing group has defined open systems as:

- A [vendor neutral] applications environment, compliant with international standards.
- Open system implementations must include interoperability with installed computing environments, and provide predictable functionality.
- Open Systems provide interfaces for interoperability or portability of applications, data, information or people.

5.2 Mid-Range Systems

As hardware technologies (VLSI, RISC, and so forth) evolve, the price/performance curve has swung to the advantage for mid-range system vendors, primarily in the low-end. Over the next three years, it is expected that price/performance changes will affect the high-end of the mid-range as well, due to the RISC technology strategies currently in formulation.

Companies are responding to fast changing market conditions such as international competition, deregulation and currency swings. Operations are being decentralized and moved closer to where the business is being conducted. If distributed computing continues to escalate, there is a greater chance of the mid-range market growing rapidly.

The Gartner Group believes that mid-range systems will be key delivery platforms in the future for client/server and end-user computing. Gartner also believes that, technologically, mid-range systems are in an excellent position to deliver the most effective PC Integration and client/server facilities.

5.2.1 Distributed/Decentralized Computing

The current trend is to distribute more of the compute power (systems) to where the computes are needed. Coupled with mid-range system compute power, client systems can supplement compute resources to the server system. These client systems benefit by the advantages mid-range systems offer, such as remote restarts, booting, backups, online documentation, and application management.

Decentralization of systems is becoming a corporate strategy for many large customers. This trend is expected to grow over the next several years. These large corporations are distributing computing responsibilities closer to where the transaction occurs. The large corporate database then tracks corporate data. Mid-range server systems are preferred as data collectors and distributors for the bi-directional data collection and distribution.

6 THE OZIX VISION - Digital's Response to this Market

6.1 The OZIX Vision

OZIX delivers a series of system software products that address a variety of needs in the open systems marketplace. In addition to providing a superior implementation of open software standards, OZIX addresses the need for compute and file servers in environments that require enhanced security and reliability. OZIX products will evolve into systems that meet the demanding requirements of production and transaction processing environments, delivering maximum performance, fault tolerance, and reliability. They will be optimized for superior availability and data integrity, providing a solid foundation for online transaction processing and other customized applications.

Over the next ten years, the OZIX provides the corporation with:

- products that complete Digital's family of open, enterprise-wide computing systems
- products that allow Digital to expand the open systems market
- an industry-leading operating system technology for the 1990s
- market leverage for Digital through introduction of technology into industry standards organization

The mission of the OZIX is to provide systems that will allow Digital to compete in non-traditional UNIX markets, particularly those that require production systems and Online transaction processing. This mission is in keeping with the mission of the Open Systems Group (OSG) group to supply a complete family of open software products, and with Digital's corporate mission to supply enterprise-wide information systems.

6.2 Customer profile - the non-traditional UNIX market

Customers who purchase UNIX systems today tend to fall into three major categories: tradition-driven, technology-driven, and policy-driven. Tradition-driven customers are interested in public-domain source code and have no interest in a vendor's added value. These customers are typified by the education market - they want Unix just for the sake of UNIX. The non-traditional UNIX market encompasses technology- and policy-driven customers described as follows:

- Technology-Driven. These customers purchase solutions that happen to have UNIX-based software platforms. They seek access to new technology that offers leadership price/performance and third-party applications. Solutions purchased by these customers are used for many applications, for example, vector processing, ECAD, retail, and financial analysis tools.
- Policy-Driven. These customers are primarily concerned with protecting their investment in data and applications. They balance the need for a stable application environment against the desire for new technology and innovation. Governments and their contractors have traditionally driven this market, but now many firms interested in protecting their applications investments are seeking UNIX solutions.

7 OZIX V2.0 Strategy

OZIX is being built as Digital's response to the demands of the open production systems market. The transaction processing computing style is of utmost importance to the success of this product, and many of the required features will be phased in over several releases of the product. Therefore, integral transaction services and database technology will be embedded in the operating system as part of the OZIX solution, part of Digital's complete solution in the Open System Commercial Marketplace.

The key to the OZIX success is the value-added functionality offered as part of the base operating system. DECwindows/MOTIF will be used extensively in areas of human interaction such as editing, mail and utilities. There will be a powerful, easy-to-use system management interface that will make the largest of system configurations (and environment) easy to manage.

Performance analysis will be used frequently and extensively to "fine-tune" the operating system as it is being built to determine where best to make major performance improvements. It is important to Digital's future to deliver industry-leading TP performance on an open systems platform.

7.0.1 Transaction Services (DECxTP)

Digital's DECxTP program includes an extensible framework that addresses open-ended application sets requiring TP technology. DECxTP provides an infrastructure of building blocks, rules, and workbenches that allow the integration of hardware and software products into a system that meets the dynamic requirements of the customer.

Increased availability and reliability requirements of products are met by the cooperative processing technology change DECxTP provides. Cooperative processing is an integrated and automated work-to-do list for an organization or function. Flexibility of modeling the business process flow allows user productivity to increase regardless of platform size, from desktop to mainframe.

DECxTP allows users to conduct their work in a cooperative fashion. The cooperative process work flow is achieved through a set of requests that are issued by several processes to a coordination point. These processes are filtered and scheduled to another coordination point until the job is done.

The DECxTP Business Transaction Model provides a framework for long-lived activities to be executed in order without the locking of resources. As a result, the level of consistency is maintained without any decrease in system performance. The new employee hiring process, as a transaction processing application, is an example of how the Business Transaction Model works - writing the job description, the signature cycle, recruiting, the job offer, acceptance of offer, and employee start date. These steps are all controlled by the framework of the Business Transaction Model.

DECxTP conforms to existing standards and promotes open systems and new TP standards as they are developed. The first implementation of DECxTP is targeted for the OZIX system.

7.0.2 Database Technology

Digital's Database Systems (DBS) group product development goals include the delivery of relational database systems and tools for TP, production and end-user applications in Digital's distributed computing environment. A secondary emphasis is to extend Digital's relational database systems with features and interfaces for scientific and engineering markets.

The Database Systems group expects to significantly expand access to the TP and commercial end-user computing markets by delivering database systems that have high performance and scalability.

Additional DBS product development goals include:

- Provide the underlying database architecture and technology for distributed information management systems. Just as network architecture and technology are the foundations of Digital's distributed computing systems of the 1980s, distributed database architecture and technology will be the foundation to Digital's success in distributed information management in the 1990s.
- Integrate Digital's heterogeneous environment and provide interoperability with competing vendors, notably IBM, through the use of an integrated database architecture. In Digital's heterogeneous environment, VMS and UNIX (ULTRIX and OSF) are of equal strategic importance. Deliver equivalent and compatible database systems for both VMS and UNIX environments on all Digital platforms.

- Achieve technical parity with the TP industry leaders (IBM and Tandem) in the areas of availability, data integrity, and reliability; gain market advantage by being a complete systems solution vendor.
- Implement secure databases that conform to Department of Defense (DoD) security standards.
- Deliver tools for the database administrator in the areas of database design and analysis, capacity planning, performance tuning, and maintenance.
- Initiate product development efforts for an object oriented database system, image database system, and knowledge-base database system.

The strategy for database runtime systems is to develop one base technology for Digital's multiple operating system and hardware platforms. The first application of this technology is RdbStar on the VAX/VMS platform. RdbStar will then be ported to other Digital supported UNIX platforms, including ULTRIX/OSF, OZIX, and various PC environments as expeditiously as possible. Digital's relational database environment will then consist of homogeneous, distributed software on heterogeneous operating system/hardware environments.

8 OZIX - Operating System Overview

For OZIX to be successful in the open production systems market, and ultimately the transaction processing segment of this market it is essential that at a minimum the system provide the following functionality:

8.1 Industry Standards

It is a primary requirement that OZIX systems be compliant with most industry standards including OSF, X/OPEN, POSIX, ANSI, and others. The OZIX will comply with most of the industry standards used that define systems as open systems.

8.2 Security

OZIX systems will be secure systems. With the proliferation of worldwide computer networks and the increasing number of security violations, customer have realized that they need more secure systems. By 1992 the United States government will require a C2 rating for all computer systems purchased. The worldwide trend is towards DoD certification at the B2 security level. By the mid-1990s both government and commercial users will require B2 (or equivalent) rating. For OZIX systems to be competitive in this market Digital *must* provide this basic feature.

8.3 Internationalization

Customers who purchase large TP systems are primarily large multinational corporations. Many of these corporations have worldwide operations. The growth of the worldwide computer systems market is increasing. The largest computer markets are the United States, Japan, and France. It is important to the OZIX to be marketed worldwide as an internationalized product. OZIX systems are built as fully international products by support of a multioctet character set (MOCS) environment. that is expected to be available at the first release.

8.4 Interoperability

Customers in the open production systems market expect that regardless of hardware or software vendor, all of their systems will have the capability of connections and exchange of data with Digital-supplied systems. The OZIX system will be built as an interoperable operating system. It is one member of the OSG open system products. These systems enable customers to create portable applications spanning Digital's open systems implementations. Applications are able to interoperate amongst hardware architectures. The OZIX system provides interoperability by offering functionality through the use of a common calling standard, a common object file format, and support of user-written UNIX-style device drivers.

The OZIX system provides a high degree of interoperability with VMS, ULTRIX, and other vendor operating systems. Functionality required to accomplish this includes DECnet, Application Integration Architecture components, and Digital's distributed system services.

8.5 Performance

The open production systems market dictates specific performance requirements in order to be a recognized player. Opportunities for the OZIX system dictates support for a high-speed database and transaction monitor that will support configurations of character-cell terminals, workstations, or both.

The following table shows performance test results for Tandem and IBM who are both considered competitors for the OZIX system in the database/transaction processing market.

Table 1: Tandem and IBM DebitCredit Benchmarks

Vendor	CPU-type	Database Software	TPS	Comments
TANDEM	VLX Guardian	Nonstop SQL V1	208 TPS	Tandem's ET-1 Benchmark [1]
		Nonstop SQL V2	208+TPS	Estimated TPS
	904E - "Cyclone"	Nonstop SQL V2	???	
IBM	3090-600S	DB2	130 TPS	Standard DebitCredit benchmark [2]
		IMS	257 TPS	
		FASTPATH	386 TPS	

[1]ET-1 benchmark tests for 90% of transaction completion within 2 seconds.

[2]The DebitCredit workload tests for 95% transaction completion within 2 seconds. This benchmark is widely accepted as a starting point for tests by many companies.

The Gartner Group predicts that by 1993 some vendors in the transaction processing market will offer complete systems capable of greater than 1000 TPS. Currently, general purpose TP systems deliver 1.2 to 2.5 TPS/MIP. The efficiency of future systems is expected to be around 5 TPS/MIP. OZIX will meet or exceed performance requirements expected in this market.

8.6 Availability

The OZIX system (hardware and software) must be highly available and reliable. File system reliability is of utmost importance in this environment. The OZIX system provides a robust file system that is fault-tolerant, high-performance, and recoverable. Catastrophic system failures are detected and recovery mechanisms ensure system and file integrity. This OZIX operating system is built with a constraint that there of less than one system crash per year. System reconfiguration and software installation occurs without service interruption to the customer.

The OZIX system requires a hardware platform that provides the high availability and fault-tolerant features expected in this market. Although software provides much of these requirements, robustness and marketability can only be achieved through a combination hardware/software solution.

9 OZIX - The Applications Requirements

The mission of the OZIX is to provide systems the allow Digital to offer customer solutions in non-traditional UNIX markets, particularly those that require production systems and Online transaction-processing capability. Layered products and end-user applications are integral elements of this solution.

There are many functions to a transaction processing system. Many of these functions need to be developed by Digital in order to have a fully integrated, high-performance transaction processing system, but, some third-party embedded products are required. Key functions are: application and transaction shells, resource managers, DECforms (front ends), development environment (DECtp workbench), system management environment, and DECintact/ACMS.

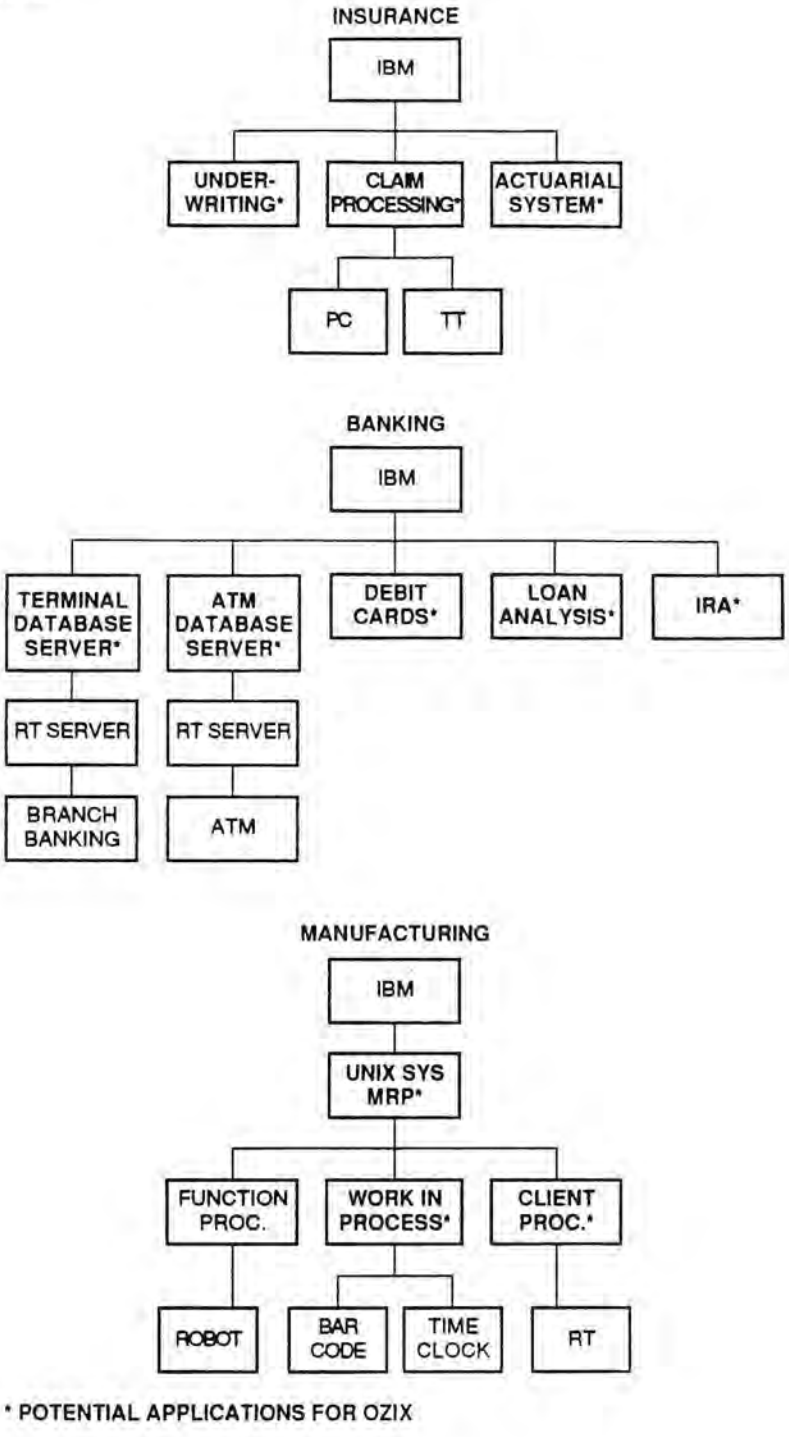
A variety of application packages are required for the TP market. For banking applications the following is needed: ATMs, TouchTone [1] phones, Videotex systems, and credit-debit cards. For insurance applications, application-processing, policy insurance, underwriting, life-insurance proposals, and claims-processing packages are needed. For broker and investment applications, stock trading, trade-processing, and client accounting packages are important. Videotex applications are also a high-growth applications area.

Network management, control, and reconfiguration applications are required for the TP market. Communication control software is also equally important to this market.

The following shows typical application profiles for the insurance, banking, and manufacturing environment. Potential applications areas for OZIX are identified with an asterisk.

[1] TouchTone is a registered trademark of AT&T

Figure 1: TP Profile



* POTENTIAL APPLICATIONS FOR OZIX

10 THE DIGITAL "WIN"

Digital has an excellent opportunity to have a leadership position in the 1990s in the open systems market. The goal of OZIX is to meet and surpass the essential requirements in gaining a significant leadership role in this market. OZIX will accomplish this goal by:

- Meeting all of the major requirements expected by customers in this market. This is measurable by using the Gartner Group ten essential requirements model.
- Offering a rich suite of applications available over all of the major market segments and platforms. This includes Digital-developed products, third-party products, and CASE tools.
- Providing the solutions where UNIX fails in the TP market in the areas of disk management, process management, memory management, data management, and fine tuning.
- Tight integration of transaction processing and database technologies.
- A hardware platform offering the latest in technological advancement that complements and adds value to the system.
- Compliance with industry standards.
- Meeting government and commercial security requirements.
- Announcing a suite of third-party and Digital applications ready to run on the OZIX platform.

APPENDIX A

BIBLIOGRAPHY

The following is a list of referenced articles used in the writing of this document.

1. *Shaping UNIX for Transaction Processing.* Ron Carnahan. March 1989.
2. *UNIX and Online Transaction Processing.* Brian Clark. April 1989.
3. *Various papers from the conference on Midrange Computing in the 1990's.* Gartner Group. June 1989.
4. *UNIX Goes Upscale.* Jeff Moad. February 1989.
5. *TP Market Sizing Document* Garth Reid, Chuck Armatys, Gary Kuba, Roy Schulte. September 1989.

October 1989

OZIX Version 1.0 Development Plan

Revision number: 1.0

At Exit from Phase 1

Issued by:

Benn Schreiber, *OZIX Project Manager (DECWET::SCHREIBER)*

Approved by:

John Gilbert— OZIX Development Manager

Roger Heinen—OSG Group Manager

Digital Equipment Corporation
Confidential and Proprietary

OZIX SYSTEMS PROJECT TEAM

OZIX Engineering Manager:

John M. Gilbert, *OZIX Engineering Manager*

OZIX Engineering Team:

Benn Schreiber, *OZIX Project Manager*
Marilyn Fries, *OZIX Development Supervisor*
Dick Funk, *OZIX Development Supervisor*
Mark Ozur, *OZIX Development Supervisor*
John Penney, *OZIX Development Supervisor*
Mike Peterson, *OZIX Development Supervisor*
Dave Snow, *OZIX Development Supervisor*
Bill Watson, *OZIX Development Supervisor*
Lu Anne Van de Pas, *OSG Compiler Development Supervisor*
Steve Jenness, *OZIX Network Architect*
Chris Saether, *OZIX ABA Architect*
Jim Schirmer, *OZIX Security Architect*
Claire Cockcroft, *OZIX Internationalization Program Manager*
Pete Benoit, *OZIX Project Leader*
Richard Brown, *OZIX Project Leader*
Sumanta Chatterjee, *OZIX Project Leader*
Jan D'Addamio, *OZIX Project Leader*
Mark Ditto, *OZIX Project Leader*
Dennis Doherty, *OZIX Project Leader*
Min-Chih Lu Earl, *OZIX Project Leader*
Debbie Girdler, *OZIX Project Leader*
Kelly Green, *OZIX Project Leader*
Jeff Havens, *OZIX Project Leader*
Brett Helsel, *OZIX Project Leader*
Oscar Newkerk, *OZIX Project Leader*
Charles Olivier, *OZIX Project Leader*
Kim Peterson, *OZIX Project Leader*
Jim Teague, *OZIX Project Leader*
Dave Walp, *OZIX Project Leader*
Charlie Wickham, *OZIX Project Leader*

OZIX Finance:

Judy Fox, *OZIX Business Support Manager*
Maggie Schimpf, *OZIX Financial Manager*
Salley Anderson-Teague, *OZIX Financial Analyst*

Product Management:

Mike Parker, *Manager OZIX Product Managers*
Kathy Appellof, *OZIX Transaction Services Product Manager*
Terry Morris, *OZIX Product Manager*

Cathie Richardson, *OZIX C & Third Party Applications Product Manager*

Manufacturing:

Rose Ramsey, *Software Manufacturing Product Manager*

Documentation:

Jim Jackson, *OZIX Publications Manager*
Liz Hunt, *OZIX Application Programming Environment Project Leader*
Marcia Aguero, *OZIX System Management Environment Project Leader*
Bill Talcott, *OZIX Software Support Environment Project Leader*
Cheryl Snyder, *OZIX Usability Engineering/Testing Project Leader*

Customer Services:

Bill Hilton, *OZIX Customer Services Systems Engineering Manager*
Thomas Siebold, *OZIX CSSE Maintainability Engineer*
LeeAnn Stivers, *OZIX CSSE Product Manager*
Myrna Harrison, *OZIX ESDP Project Leader*

TABLE OF CONTENTS

FIGURES

TABLES

Preface

The OZIX Version 1.0 Development Plan describes the strategy and tactics followed to design and develop Version 1.0 of OZIX, a member of the ULTRIX family of operating systems.

Change History

Date	Issue #	Description
October 15, 1989	0.1	Preliminary OZIX Version 1.0 Development Plan
October 31, 1989	1.0	Incorporate review comments

Date of Printing:
31-OCT-1989 16:59:32.41

1 Executive Summary

The production systems environment is divided into two distinct segments, proprietary and open systems. Digital must have products for both types of customers. Currently, VMS answers the need for proprietary production systems where the primary competition is IBM MVS large systems and the AS/400 in the mid-range. However, in the open systems segment Digital currently does not offer what customers perceive to be a commercially focused system which can compete with IBM's AIX, AT&T's System V, HP's UX, and Pyramid's OSx.

Digital has a window of opportunity to establish itself as a major vendor in the open systems production market by offering industry-leading open systems products. This document defines a plan to build an industry-leading product targeted toward the open systems production environment.

OZIX is a project to design and deliver a simple, extensible, hardware-independent operating system, adding a commercial, high-end system platform to the ULTRIX family. OZIX integrates modern technologies for distributed systems, fault tolerance, and data integrity, in conjunction with a high-performance I/O design. Its capabilities are implemented over a series of releases and are composed of system elements from other Digital groups (such as database and transaction processing software, compilers, tools, and utilities), in addition to selected elements from third-party vendors.

The OZIX Version 1 product is a high-performance production system targeted at the open systems market. Transaction processing technologies are built into Version 1 and utilized by system components to obtain a high level of performance and reliability. OZIX Version 2 delivers a robust, distributed transaction processing system with a complete set of transaction functions and advanced database technologies available to application developers.

This plan defines the goals, strategies, tactics, and deliverables of the first release of the product.

2 Product Goals

The OZIX project provides an operating system for the open systems production environment that embodies the following goals:

- **Superior implementation of open systems standards**

  Many customers developing their 1990s computing environments are demanding system solutions based on open systems standards. Customers perceive that open systems provide them with a wider application base. Customers are also looking for ways to become more independent from the hardware vendor; customer and third-party applications built to open systems standard interfaces are believed to provide such independence.

  The OZIX product delivers an operating system and environment in which the basic system concepts embody the models of process management, file access, and system calls defined in the open systems standards bodies, such as POSIX, X/Open, and OSF AES. Because these key concepts are the "native" mode of the system, OZIX delivers higher performance to standards-conforming applications than conventional operating systems with layered open systems interfaces.

  OZIX is superior to existing UNIX implementations due to the integration of transaction processing technologies into the basic system design for increased robustness and recoverability. The OZIX system design provides a high level of system security (National Computer Security Center (NCSC) B2 rating) and robustness with open systems interfaces. Existing UNIX implementations must be substantially redesigned and reimplemented to incorporate these concepts into their existing architectures.

- **High level of security**

  There is a world-wide trend toward requiring more secure systems. It is expected that by the mid-1990s, the U.S. government will require B2 security, with similar requirements following in commercial sectors.

  OZIX is designed and implemented to deliver a product that can be certified at the NCSC B2 security level. The B2 level of certification assures that a controlled development process and good software engineering techniques are used, helping to create a stronger, more error-free system.

  OZIX Version 1 delivers the fundamental system architecture and methodology required for eventual certification. Subsequent releases will complete the implementation of the required functions to support full certification. The security capabilities of the system may be customized to suit the particular security requirements of individual customer needs, such as those of commercial installations.

- **Robust, high-performance, recoverable file system**

  In the production systems market, data integrity must never be compromised. Loss of data can be fatal to the business. Should the system fail, for whatever reason, the time to recover the file system to a consistent, stable state, containing all previous updates, must be minimal, to ensure the highest level of application availability.

  The OZIX file system provides a high degree of availability and data integrity by utilizing transaction processing techniques to ensure robustness and minimize recovery time. The same techniques greatly reduce the number of disk writes necessary to reliably commit storage updates as compared to careful writing methods. This greatly improves (reduces) the latency of storage updates for applications.

  The OZIX file system supports the POSIX 1003.1 file system interface, with additional resource management services to efficiently support database systems in a production environment.

- **I/O performance and connectivity**

  One of the most critical components of a high-performance transaction processing system is the I/O subsystem, which must address requirements of connectivity, throughput, and low latency.

  OZIX is designed to deliver the highest mass storage and network throughput per unit of computational power, as well as minimizing data access latency. Disk striping, advanced storage management, and high I/O connectivity are implemented to contribute to superior I/O performance. The I/O system is designed in such a way as to take advantage of newer I/O configuration tactics, such as XOR stripe sets, as they become commercially viable. Special attention is paid to thorough design and testing, to ensure that the reliability and integrity of the user's data is preserved.

  The Attribute Based Allocation (ABA) architecture couples the file system and mass storage I/O system in an elegant and innovative way. ABA is the basis for dynamic storage system tuning by providing online storage migration primitives. These primitives are used to continually balance the use of mass storage system components for maximum system performance.

- **Fault tolerance, reliability, and availability**

  Production systems applications are mission critical. System down time can be incredibly costly. The ability to remove failing elements from active system use, while keeping the system and applications available, is critical.

A system-wide architecture for fault detection, fault recovery, and error logging is a key feature of the OZIX system. These components work in conjunction with the hardware capabilities to provide a high degree of system and application availability.

- **Internationalization**

  Many corporations today have world-wide operations. A necessary component of consistent enterprise-wide data processing for these concerns is a system that supports world-wide localization, and tailorability to various language-specific input and data manipulation requirements.

  OZIX products are designed to be sold in international markets. Critical components, such as the file system, application programming interfaces (APIs), and the terminal services, are designed and implemented to accommodate compound strings and multi-octet characters (MOCS) for textual representations. These features, which are integrated into OZIX from the outset during design, ensure that Digital, customers, and third-party developers are able to develop applications for world-wide delivery or tailored for specific cultures and local languages.

- **Reduced cost of ownership through advanced system and network management**

  One of the significant costs of ownership of large production systems is the staff required for system administration. Customers are looking for ways to significantly reduce their operations cost while increasing the overall effectiveness of the applied resources.

  OZIX management capabilities are designed and implemented with the goal of maximizing the operations staff productivity and lowering the overall cost of system management.

- **Scalability**

  OZIX is designed to efficiently support large physical memory configurations, a large number of processors, terabytes of data on hundreds of disk drives, large numbers of processes, and very large network configurations.

  The modular design of OZIX allows system capabilities to be tailored to the configuration on which the system is running, so that valuable physical memory is not consumed by system code supporting hardware not in the configuration. Dynamic system data structure allocation facilitates adding additional hardware to the configuration without rebuilding the system.

- **ULTRIX Familiness**

  The compatible set of ULTRIX base systems provides customers and developers with distributed interoperability and a high degree of application portability.

  OZIX is an integral member of Digital's ULTRIX family, and shares a family of capabilities with all ULTRIX products:

  — A common set of application programming interfaces (OSF AES, POSIX 1003.1, XPG)

  — A common set of application development tools, utilities, and commands (POSIX 1003.2, XPG)

  — A common software architecture on like processor architectures (calling standard, object file formats, and so on)

  — A unified network and distributed systems architecture

  — A unified system and network management architecture

  — A common core of documentation

- **Portability**

It will be necessary to move the OZIX system to various hardware platforms, as Digital's business requirements change. The system is constructed in such a way as to provide high performance on target hardware platforms, yet it is designed and implemented in such a way as to be readily portable.

OZIX is designed to readily identify and localize the hardware-dependent aspects of the system to facilitate portability.

3 Product Description

The OZIX project delivers a robust, secure, high-performance operating system which is positioned as a member of Digital's ULTRIX family of systems. It is expected that the product name will reflect the ULTRIX familiness; however, the specific external product name will not be determined until Phase 2.

3.1 Deliverable Functions

The primary deliverable components of the OZIX project are discussed in the following sections.

3.1.1 Open Systems Application Programming Interfaces

The application programming interfaces (API) implement the standard programming interfaces to the operating system, including OSF AES, POSIX 1003.1, and X/Open XPG. These APIs provide the programming interface between user applications and the OZIX executive.

Additional APIs provide management interfaces to the various subsystems, support for fully internationalized applications, and other unique OZIX added-value capabilities.

3.1.2 OZIX Executive

The OZIX executive is comprised of a set of modular units, or subsystems. These subsystems implement the functionality required to support the application programming interfaces, as well as the memory management support, security check and security auditing, I/O drivers and support routines, and fault management and recovery. The OZIX design places each subsystem in a unique address space to ensure robustness, modularity, and fully address the security and integrity requirements of the B2 level of security.

The executive subsystems are supported by a small *nub*, which provides the primitive functions necessary such that the remainder of the executive can be efficiently implemented in subsystems. These functions, which are not user-visible, consist of low-level thread scheduling and synchronization, CPU management, fault detection, interrupt dispatch and control, and cross-subsystem procedure call support.

Throughout the executive, including the file system directory structures, the text for all objects that can be named is stored as compound strings. Compound strings provide support for multi-octet characters as well as other character information, such as writing direction.

The executive is being designed in close collaboration with the HPS/DECxTP engineering group to ensure that the design addresses the critical requirements of a high-performance transaction processing system.

3.1.3 Robust, Recoverable File System

The file system consists of a set of subsystems that implement the emerging Attribute Based Allocation (ABA) storage architecture. In this architecture, the file system may be viewed as being implemented in three layers:

- File name subsystem

  The file name subsystem implements a generic set of file and directory services, such as open, read, write and close, which may be called by other OZIX APIs. The file name subsystem also provides:

  — Mount point services

  Mount point services are utilized to mount objects other than local ABA files (remote directory trees via NFS, for example) in the local directory hierarchy.

  — Create and lookup routines for pipes and special files

- Dataspace subsystem

  The dataspace subsystem implements a virtual interface from file name subsystems with a mapping onto mass storage media. Each dataspace has specific associated attributes, such as size and access characteristics. The dataspace subsystem provides a generic read/write service interface which is used by the file name system and database systems, as well as a management interface for use by OZIX resource management.

- Container subsystem

  The container subsystem presents mass storage devices to the dataspace subsystem as a logical representation of a device, or container. Several types of containers are implemented:

  — Base containers

  Base containers represent physical mass storage devices, such as disk drives.

  — Subcontainers

  Base containers may be partitioned into multiple subcontainers, with each subcontainer representing a portion of the entire container.

  — Compound containers

  Compound containers represent more complex virtual abstractions, such as disk striping and shadowing. Additional compound container types will be developed over various releases of OZIX to address future customer requirements and take advantage of future mass storage capabilities to provide enhanced performance, robustness, and reliability.

3.1.4 Networking Components

The OZIX networking components are designed and implemented in conjunction with the executive architecture, and take full advantage of the performance and security capabilities. The fundamental components of the OZIX networking system include:

- Internet network subsystem

  The Internet network subsystem implements the full suite of Internet protocols, including:

  At the application layer:

  — Routing database management daemon

  — Berkeley Internet Name Domain service (BIND)

— X/Open Transport Interface (XTI)

— Socket interface

— The r* utilities (rwhod, rsh, rcp, and so on)

— Network time protocol (NTP)

— The internet remote terminal and file transfer services

At the network layer:

— Internet Protocol (IP)

— Internet Control Message Protocol (ICMP)

— Internet Group Management Protocol (IGMP)

— Address Resolution Protocol (ARP) (includes Reverse Address Resolution Protocol (RARP) and Proxy ARP)

At the transport layer:

— Transmission Control Protocol (TCP)

— User Datagram Protocol (UDP)

- DECnet/OSI network subsystem

  OZIX DECnet/OSI is an end-node implementation of DECnet/OSI Phase V, with support for multiple physical links on Ethernet (multilink) for improved throughput and availability. The DECnet/OSI components include:

  At the application layer:

  — File Transfer and Access Management (FTAM)

  — Virtual Terminal Protocol (VTP)

  — DECnet/OSI Copy Program (DCP)

  — The d* utilities (dlogin, dcp, and so on)

  — Distributed Name Service (DNS) clerk

  — Digital Time Synchronization Services (DTSS) client

  At the session layer:

  — DNA Session

  — OSI Session

  At the transport layer:

  — OSI Transport Protocol (TP4)

- Remote Procedure Calls (RPC)

  Remote Procedure Call capability is provided by an implementation of DECrpc V1.

- NFS

  OZIX provides a high-availability, high-performance implementation of Sun Microsystems' NFS V2.0, that provides remote file client and server services, as well as remote lock and status managers.

3.1.5 OZIX I/O

OZIX I/O is designed and implemented in conjunction with the executive, and takes full advantage of the advanced performance capabilities. OZIX I/O provides state-of-the-art performance by exploiting the parallelism of symmetric multiprocessing and multithreading.

Areas addressed within the design of OZIX I/O include:

- Minimal lock contention
- Buffer management techniques that avoid or minimize data copying
- High levels of I/O connectivity for maximum parallel operation
- Well-designed management of I/O components eases management requirements in large configurations
- Modular design facilitates the addition of support for new peripheral and adapter technologies

OZIX terminal services support LAT terminals in addition to the various network terminal services. The terminal services are designed to ultimately be implemented independently by various supporting engineering groups, and, when delivered, support multiple character sets concurrently on a single system. Engineers at the Japan Research and Development Center in Tokyo are participating in the initial design of the terminal services and the implementation of the Japanese input methods and character set support.

Devices supported by OZIX I/O at FRS include Ethernet, Computer Interconnect (CI) for access to high-performance mass storage on DSA disks, magnetic tape, and CDROM.

3.1.6 System and Network Administration

The OZIX system administration design addresses the requirements of both production systems and management of B2 security policies. The OZIX system administration structure is designed into each component of the system in an integrated fashion.

The OZIX system administration structure is based on an object-oriented methodology in conformance with Digital's Enterprise Management Architecture (EMA). All tasks performed by system administration are performed on manageable objects that represent users, devices, processes, networks, and so on. As objects, these components are identified by their specific attributes, operations, and events.

A management backplane is used to interconnect manageable objects to each other and to management applications. The same EMA-compliant management backplane is used by both OZIX system administration and network administration providing a common set of services across interconnected systems.

3.1.6.1 Management Applications

Management applications request management operations to be performed. The request for an operation is made by the management application, through the management backplane, to the object. Objects interact indirectly with management applications and have no knowledge of the management applications.

The types of management applications are unlimited. Some management applications are simple user interfaces using command line and/or DECwindows technology. Other management applications may be more complex decision-making applications that issue management operations based on a set of rules. As for the object being requested to perform some operation, the type of management application is immaterial.

OZIX provides a set of user presentations and a limited number of decision-making applications for resource and fault management.

3.1.6.2 User Presentations

A user presentation is a management application providing an interface that system and network managers use to manipulate manageable objects. A set of user presentations is implemented to provide a variety of interfaces for use through character-cell terminals, shell command scripts and DECwindows terminals.

3.1.7 Performance Tools

Low-overhead event detection and sampling services are key requirements for systems targeted toward computer center and mainframe-oriented facilities where system tuning, resource allocation, and capacity planning are paramount considerations in delivering enterprise-wide services.

OZIX provides a full set of performance measurement and application characterization tools. More significantly, OZIX incorporates an instrumentation strategy that allows higher levels of software to obtain performance data in a well-structured, consistent, and low-overhead manner.

The OZIX instrumentation strategy is implemented in an Integrated Data Collection Facility (IDCF). The IDCF collects performance data from the various manageable objects via the management backplane. The performance data is made available to higher levels of software via a robust and well-defined set of data collection services.

By making these services consistent with emerging standard interfaces, or in the absence of standards making them available to third parties and other application developers, OZIX will evolve into an operating system with one of the broadest ranges of high-quality performance measurement, analysis, and resource management tools in the industry.

3.1.8 Commands, Utilities, and Libraries

The basic command set provided with OZIX consists of the commands included in the OSF/1, X/Open, and POSIX 1003.2 standards. This set of commands includes a subset of the SVID commands because a portion of these were used in producing these standards. In addition, a subset of the BSD commands available with ULTRIX are provided where these commands are the easiest, most reasonable, most efficient, or possibly the only way to accomplish the required function.

The OZIX libraries include the components that give the application designer a known programming environment that complies with available open system standards, as well as the richness expected of a modern operating system. These include:

- Standard C Libraries

  These are compliant with:

  — OSF/1

  — X/Open (XPG.3)

  — POSIX 1003.1

  — ANSI C

  These libraries include the Concert Multithread™ Architecture (CMA) application level functions and are thread safe.

- Compound String Libraries

  The Compound String Libraries provide the application interface for internationalization programming support. These libraries implement compound string manipulation and conversion functions and the compound string interfaces to input/output routines.

- ULTRIX Libraries

  It is a major goal of OZIX to maintain interoperability and portability of applications with the ULTRIX family of systems. This set of libraries is a subset of the supported libraries commonly found on ULTRIX that are used in modern applications. The exact content of this set can be found in the *OZIX Libraries Functional Specification*.

3.1.9 Diagnostic Monitor

The Online Diagnostic Monitor (ODM) provides the user interface and the controlling mechanisms for running online diagnostic programs. Online diagnostic programs are used to test or exercise system hardware and/or software without taking the system offline, thus causing a minimum amount of disruption to system availability. ODM provides:

- Value to customers as a sanity check and verification of the system configuration validity, and as a component of the overall customer acceptance testing
- A common operating environment and user interface for all online diagnostic programs, including the Online System Exerciser (OX)
- The ability to run multiple diagnostics, sequentially or concurrently
- Scheduling tests within each diagnostic
- A common set of services for the diagnostics tests
- A common user interface for all diagnostics, from a character-cell terminal and a DECwindows environment
- Internationalization of diagnostics

The Online System Exerciser is an integral part of ODM. It applies software loads to the system, exercising it through the hardware/software interface. The use of OX provides:

- Exercisers for disk I/O, file system, network I/O, CPU, memory management, and tape I/O
- A method by which the I/O, network, file system, and performance groups can load the system in a controllable manner for their own testing
- A way to stress test the OZIX system software and hardware during development to aid in discovering software errors
- A tool for use by manufacturing to perform Stage III checkout and burn-in testing
- A tool for use by Customer Services to verify the installation of new hardware or software at the customer site and to troubleshoot system problems

4 Design Strategy and Tactics

The overall design strategy for OZIX is to utilize contemporary computing concepts, such as an innovative base system architecture and mass storage model, and develop detailed designs which incorporate these concepts.

Detailed functional and design specifications, along with thorough design reviews, ensure that the specifications meet the product goals, as well as:

- Provide a solid basis for the development of technical and user documentation
- Provide a set of technical documentation for future OZIX engineers
- Ensure the proper methodology used to address NCSC B2 security requirements

The OZIX design is driven by a vision of computing focused several years in the future and incorporates many contemporary design concepts. The initial products resulting from the OZIX project are delivered as single computing elements in a network, interoperating via "traditional" distributed techniques.

However, the fundamental system designs incorporate structures that ease the task of extending the system in the direction of the long-term computing vision. These extensible design concepts can be seen in the memory management and file system designs, two fundamental components of the future distributed computing environment.

4.1 Design Methodology

The overall design methodology addresses the areas of functional specification, interface specification, detailed design, implementation, and testing:

- Functional Specification

  Functional specifications are a high-level summary of what a component does. The functions described map to the interface specification for details, to the detailed design for implementation, and to the code that actually performs the functions. These specifications are taken through a rigorous review process that includes a high-level broad review, and a detailed review by the primary consumers of the functions.

  Comments and discussions from the interface specification review, as well as interface and detailed design reviews, are captured online for future reference, and to satisfy the requirements for NCSC B2 certification.

- Interface Specification

  The interface specification provides the details of how a component is used. It contains sufficient information to allow others to write code that uses the component. Because of the extensive dependencies typical of this level of specification, changes to the interface specification are relatively widely reviewed.

- Detailed Design

  The detailed design specification is a description of how the component performs the functions defined in the functional specification. The purpose of the detailed design is to allow others to understand the internal design and workings of a component.

- Implementation

During the implementation phase, rigorous code reviews ensure the quality of the code under development. The project leader and developer are responsible for ensuring satisfactory code review coverage, except that *all code belonging to elements of the trusted computing base (TCB) is reviewed.*

- Testing

  Quality and testing are an integral part of the OZIX development. Testing is conducted during development even before the operating system can run on hardware by use of a simulator. It continues though field test, and beyond the product FRS. It is conducted by all levels of development from unit test of a module or component, up through system integration, to layered product and third-party software testing.

  The product goal is to test at least 80% of all code in each component of the operating system. The NCSC B2 security classification requires that components in the TCB have test coverage close to 100%.

  For an overview of testing refer to the *OZIX Software Quality and Testing Strategy*, and for a more detailed understanding, see the *OZIX Software Test Plan.*

4.2 Performance Measurement and Reporting

The performance of all OZIX-based systems is reported in terms of *transaction response time*, *system throughput*, or both. The specific tests from which the metrics are derived include various de facto standard performance tests and internally-developed customer-environment workload tests. Where appropriate, these metrics are obtained under a variety of conditions (e.g., varying system parameters) so that the ability of OZIX-based systems to deliver performance under different conditions can be properly assessed.

To date, five representative customer workloads have been selected to apply to OZIX. They were selected principally on the basis of their I/O profile, and secondarily on the application mix. The customers from whom the workloads were derived are:

- Citibank
- DuPont
- USAA
- Batelle
- Digital

To apply these to OZIX, we abstract the workloads and emulate them using a commercial Remote Terminal Emulator (RTE) package such as Benchmaster.

Since vendor performance claims are viewed with great skepticism, for each release of OZIX we will produce a set of performance reports obtained from organizations external to Digital or, where that's not practical, Digital will conduct these tests audited by an external agency. These tests include, but are not necessarily limited to the following:

- An audited DebitCredit Benchmark†.
- Results obtained from The Neal Nelson Commercial Benchmark Suite. To be conducted by Neal Nelson and Assoc., Chicago, Ill.

† As described in Wright L., *et al*, **Digital's DebitCredit Benchmark: Methodology and Results**, Digital Equipment Corporation, Marlboro, MA, May 1989

- The AIM UNIX benchmark suite results (Suite III). To be conducted by AIM Inc., Palo Alto, Ca.

Finally, a broad set of market-specific system performance tests are conducted and reported by the various PBU and PMG system engineering groups for each release. More information on these market-specific plans is being developed and is detailed in the forthcoming *OZIX Performance Testing Plan*.

5 Development Strategy

Interface specifications, design specifications, and implementation are accomplished following the procedures detailed in the document *OZIX Software Development Procedures*, which describes the detailed procedures that must be followed to ensure the delivery of a high-quality product meeting the NCSC B2 certification requirements. These procedures can only be compromised at the risk of endangering B2 certification.

OZIX is designed to be as independent of its hardware platform as is technologically reasonable. The benefits of this approach are twofold:

- OZIX can be easily retargeted to other hardware platforms
- The OZIX development schedule is less perturbed by fluctuation in hardware schedules or commitments

The second benefit is realized by integrating advanced simulator technology with the use of conventional hardware platforms. Initially, OZIX is developed using a 32-bit hardware simulator‡ and an ISIS-based (dual R3000 CALYPSO/XMI) hardware platform. DECstation 3100 workstations provide the computational power required for reasonable response times when utilizing the simulator for debugging, testing, and performance measurements. The DECstation 3100 workstations utilize ULTRIX timesharing systems for remote file services and as the repository for OZIX sources.

Since the FRS target hardware platform, a MIPS R4000/XMI2-based system, is not be available until relatively late in the development cycle of OZIX Version 1, OZIX is retargeted to this platform initially by using a customized software simulator. The simulator provides a full multiprocessor R4000 execution environment.

This strategy minimizes the risk to the OZIX schedule imposed by the target hardware platform deliverables. It also shortens the OZIX development cycle since the ISP, cache manipulation routines, locking, and other architecture-specific portions of OZIX can be thoroughly tested and debugged in advance of hardware availability.

5.1 Development Training Requirements

During OZIX V1 development, training is required for:

- NCSC reviewers–As the NCSC security review process is started, we are required to provide training for the reviewers.
- Field test support training–We are required to train the personnel that support OZIX field test sites.

Because of the timing of these training requirements, the development of the training material is done by OZIX engineering, utilizing the engineering functional and design specifications.

‡ MIPS' Sable that we have customized for software performance and quality testing

5.2 Field Test Strategy

Field test is the engineering development process by which the total hardware, software, and support system is exercised. It provides a generally friendly environment in which the system is utilized heavily in non-mission critical situations to shake out problems not detected or impossible to stimulate in the in-house environments.

Field test sites are selected based on criteria specified by engineering, to fulfill specific needs such as applications used, hardware environment, and so on. We expect to have field test sites in the U.S., Europe, and GIA, to ensure that OZIX receives testing appropriate to ensure the product goals and quality. Logistics problems with non-U.S. field test sites require extensive help and cooperation from groups within the company to facilitate a successful and timely field test.

The feedback loop, typically through a Quality Assurance Report (QAR) system, is implemented such that there is no additional time or levels of bureaucracy introduced between the submission of the problem report and its receipt by engineering, permitting engineering to respond promptly to the reports.

6 Product Configurations

To ensure the highest FRS product quality and performance, the FRS product concentrates support on a single processor type, consisting of 1 to N processors. OZIX supports up to 32 processors in a symmetric multiprocessing system. The configuration supported at FRS is likely smaller, gated by the hardware configurations available, and Digital's ability to test and verify OZIX performance on large-scale configurations.

OZIX supports a variety of memory and mass storage configurations on this system platform. Once the FRS hardware platform details have been resolved, additional information will be available on supported product configurations.

Subsequent versions of OZIX will support additional hardware platforms and I/O devices, based on product requirements.

7 Schedule of Product Deliverables

OZIX is developed through a series of carefully planned and orchestrated baselevels. System builds are done on a regular basis throughout the development of the system, and the baselevels provide integration points for development, and project progress checkpoints for project management. As the V1 system approaches maturity, the baselevels provide information for cooperating and dependent engineering groups as to the OZIX schedule and content.

The schedule is evaluated at each baselevel to determine overall progress toward the V1 product. This is especially critical due to the potential schedule impact resulting from the B2 development methodology.

The following tables describe the major content of the baselevel schedule which is followed to develop OZIX. These baselevels were defined using a methodology which primarily focused on the OZIX development requirements, such that underlying required functions are in place prior to being utilized by higher-level functions.

Application items which must run on OZIX are flagged with "#". Note that most of these are first built on ULTRIX.

Table 1: BASELEVEL 1 - Core Functionality

Deliverable	Comments
	Executive
Memory Allocation (physical)	Quota-related and virtual to physical translation routines stubbed. allocate, deallocate, and aligned allocation needed.
Create/exit threads	
Nub Services	Events - init, set/clear, read (synchronize and notification events) Mutex - init, init shared, set, release Wait services - single and multiple Queue - init, insert head, insert tail, remove head, remove tail, read Timers - init, set, read Interrupt services Memory interlock mechanism
Subsystem Services subset	Packages and gate crossing without address spaces support
Exception/condition Handling	Includes Message and Status Value support
Primitive Debug Server	
Extended Initialization entry point	Testing hook
Common IO support	IO synchronization and completion
Linked lists, byte copy, compare, and zero	
Bitmap	Create, delete, clear all bits, clear/set bit position, find free, check bit
Argument list encoding/decoding	Item list/TLV substitute
	Application
Ladebug	In support of nub debug

Table 2: BASELEVEL 2 - Basic I/O including Console

Deliverable	Comments
	Executive
basic Ethernet driver	Transmit and receive
Basic base Container services (MSCP, SCS, CI subsets)	Data transfer
Basic Console tty port driver	
Primitive tty class driver	
Primitive Event Dispatcher	Local console logging
	Application
	No new application code

Table 3: BASELEVEL 3 - Basic executive including network services, process, and read-only file system

Deliverable	Comments
	Executive
OSF API skeleton	Only the system traps
Subsystem support complete	Includes gates, trumps, subsystem load/unload. Interdependent with MM
enhanced Debug server	Supports subsystem debug
Nub complete	MM dependency. Also include cross address space copy support.
Memory Management (MM) subset	TLB fills and image file page-in support (includes primitive segment manager)
Process subset	Image file load and mapping
Dataspaces subset A	Read-only dataspaces with file system support
compound Container services subset	Striping, shadow, etc.
basic LAT services	Data transfer on virtual circuit and slot
basic Network services	Virtual circuit creation/teardown, datagram creation, data transfer without error detection and recovery, all without API support
Data Collection Facility subset	
Sun RPC	Used by NFS
System Mgmt Agent subset	Includes object interface used by subsystems to register objects and action routine handlers, and agent interface for protocol engines.
	Application
DB for system management	May be INGRES for V1.0
ODM Subset A	Enough to run OX Subset A
OX Subset A	OX manager(subset), OX interface, disk exerciser (raw file interface only)
Ladebug	Supports subsystem debug

Table 4: BASELEVEL 4 - Primitive application environment including systems mgmt, read/write file system, remote file access

Deliverable	Comments
	Executive
Queue Management	Trusted queue mechanism
MM complete	Full paging
OSF API subset A	I/O subset (open, close, read, write) and basic network services
basic Security w/o event logging	
basic TTY services	
Ethernet subsystem complete	
Event Dispatcher B	local file posting

Table 4 (Cont.): BASELEVEL 4 - Primitive application environment including systems mgmt, read/write file system, remote file access

Deliverable	Comments
	Executive
Dataspaces subset B	Create and extend files, non-recoverable write-thru writes, ACL support
NFS subset A	Client side
	Application
I18N program library	20%
C library	Full w/OZIX routines on PMAX
Implementation Independent Kernel Interface (IIKI)	On PMAX
DB for system management	Primitive subset
Non-network management services	Requires DB, single-threaded
Management Information Repository (MIR)	Requires DB, only SPC protocol handled, single-threaded
Basic USER manageable object	
DECwindows storage manager interface	Basic
ODM Subset B	
OX Subset B	File exerciser, memory exerciser
#C library	Full library on OZIX - built on skeleton API
DECrpc	

Table 5: BASELEVEL 5 - OZIX Boot and minimal shell, recoverable file system

Deliverable	Comments
	Executive
OSF API subset B	Child process creation (fork/exec)
Process complete	
OZIX Boot	
base Container services complete (MSCP, SCS, CI complete)	
Dataspaces subset C	Recoverable dataspace with F.S. support
Device special file support	
Logging and recovery subsystem	
Storage mgmt subset A	
LAT complete	
Bus Scan subsystem	Needed for I/O Configuration Mgr
generic TTY complete	Includes MOCS
NFS subset B	Server side

Table 5 (Cont.): BASELEVEL 5 - OZIX Boot and minimal shell, recoverable file system

Deliverable	Comments
	Application
I18N program library	40%
Port standard commands	40% on PMAX
OZIX specific commands	ps, etc. on PMAX
DB for system management Subset A	
Management Command Language (MCL)	
DOMAIN manageable object	
DECwindows storage manager interface to DOMAIN	
Resource/fault management Subset A	
#Minimal shell	
#Error logging	
#Pixie Subset A	"kernel mode" coverage
#Prof Subset A	Process pixie subset A data

Table 6: BASELEVEL 6 - Manageable system including IPC, tape, Japanese TTY

Deliverable	Comments
	Executive
TTY complete	Includes pseudo ttys and JTTY (japanese)
I/O Configuration Mgr	
TMSCP subsystem	
special file interface to TMSCP and MSCP	
compound Container complete	
Dataspace migration	
Storage mgmt subset B	
Network complete	Complete API support
UNIX Domain sockets	Socket based IPC
Pipes	
Event Dispatcher complete	
NFS complete	

Table 6 (Cont.): BASELEVEL 6 - Manageable system including IPC, tape, Japanese TTY

Deliverable	Comments
	Application
I18N program library	60%
Port standard commands	60% on PMAX
Pixie Subset B	"user mode" coverage
Prof Subset B	Anything extra needed for pixie Subset B
#Port standard commands	~20% on OZIX
#DB for system management Subset B	
#MCL	Handles variable syntax
#MIR	Handles standard management protocols
#DECwindows storage manager Subset A	
#Error log report generator	
#Resource/fault management Subset B	Also port to OZIX
#Instrumentation Collector Subset A	

Table 7: BASELEVEL 7—API level

Deliverable	Comments
	Executive
OSF API complete	
Crash dump writer	For system Dump analyzer
Data Collection Facility complete	
Storage mgmt subset C	
	Application
I18N program library	80%
Port standard commands	80% on PMAX
#Port standard commands	~40% on OZIX
#Ladebug	On OZIX, also remote from ULTRIX
#Crash dump analyzer	Need crash dump writer
#Full USER manageable object	
#Full GROUP manageable object	
#Special file support for /etc files	
#DECwindows storage manager Subset B	
#Full resource/fault management	
#Full instrumentation collector	
#Full ODM	Also port to OZIX
#Full OX	Full manager, network, tape, CPU exercisers, disk exerciser with container interface

Table 8: BASELEVEL 8 - OZIX V1.0 Fieldtest System

Deliverable	Comments
	Executive
	No new components
	Application
#I18N program library	100% on OZIX
#Port standard commands	100% on OZIX including C shell and Sys V Bourne shell
#C Library	Full, thread safe, CMA
#Miscellaneous Libraries	
#Shared library support	
#Linker	ld
#Librarian	ar
#Linker utilities	ldutils
#Accounting	Ala UNIX
#Mail	
#DECprint	
#IIKI	
#Full DB for system management	INGRES??
#Standard System Administration Commands	adduser, removeuser, etc.
#Full Management services	Network support
#Full system install	setld, RIS
#Full DECwindows storage manager	
#Full pixie	
#Full prof	
#sa*	

7.1 OZIX V1 Schedule

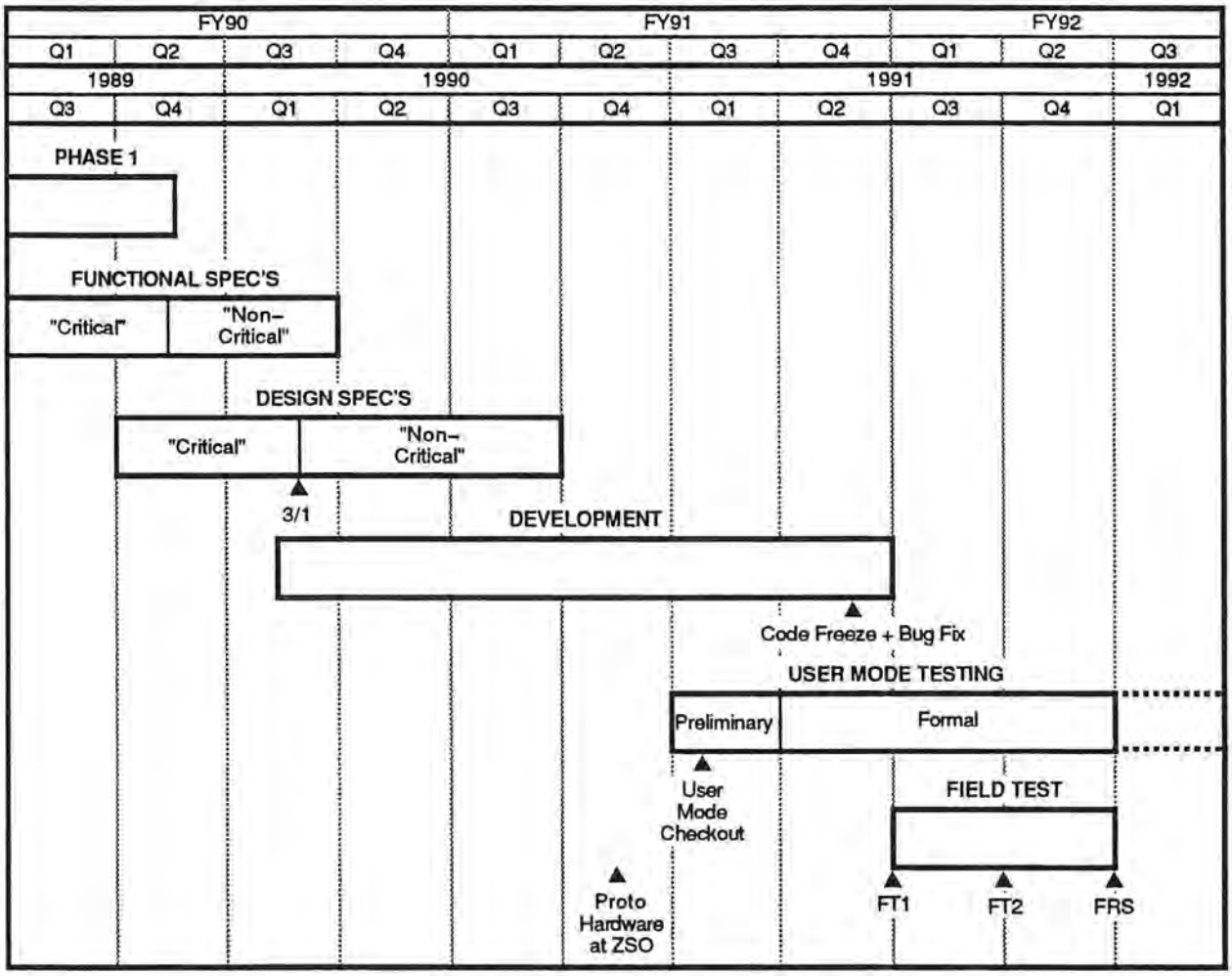
The OZIX baselevels are developed on the following schedule:

- BL1: 29-June-1990
- BL2: 17-August-1990
- BL3: 5-October-1990
- BL4: 3-December-1990
- BL5: 18-January-1991
- BL6: 8-March-1991
- BL7: 18-April-1991
- BL8: 7-June-1991
- OZIX V1 fieldtest 1 start: 1-July-1991
- OZIX V1 fieldtest 2 start: 1-October-1991

- OZIX V1 to SDC: 20-December-1991

The overall OZIX V1 development schedule is presented in the following diagram. The distinction between "critical" and "non-critical" functional and design specifications is made based on the relative position in the overall system development of the component. Components that comprise the fundamental lowest layers of the operating system, and are therefore required for subsequent development, are "critical".

Figure 1: OZIX V1 Development Schedule



8 Technical Risks and Dependencies

The OZIX project has many risks and dependencies. This section discusses those risks and dependencies where there is believed to be a significant uncertainty as to their timely resolution.

The following project risks and dependencies have been identified:

- **RISK: Design and Implementation of New Technology to a Schedule**

The OZIX product incorporates new technology in several dimensions:

- B2 security
- Base system architecture
- Mass Storage architecture (ABA)
- Internationalization pervasive in the operating system
- Advanced system management
- Acceptably fast system and application restart/recovery

The technology in many of these areas is new and unproven; delivery of the sum of these capabilities results in an industry-leadership product. However, there is some inherent risk in developing new technology to a schedule.

- **RISK: Impact of B2 Security Requirements on Development Schedule**

  Although most of the requirements on the development methodology are simply good engineering practices, many projects tend to ease off on the methodology during development when schedule pressure is encountered. That strategy is unacceptable for OZIX, as it compromises the OZIX B2 certifiability.

  CONTINGENCY PLANS: We will slip the schedule to practice the development methodology to ensure B2 certifiability.

- **RISK: Hardware Platform Configuration and Schedules**

  There are no *formally committed* schedules and plans for hardware platforms based on the R4000 and/or EVAX. Until schedules (including specifications and prototypes) firm up, there is uncertainty in the OZIX schedule.

  Significant change in the platform I/O strategy may impact the schedule due to the changes required in I/O drivers.

  In order to minimize kit testing, increase reliability, and meet the security requirements for read-only distribution media, OZIX requires that the hardware platform include an on-board CDROM reader. (This is consistent with Digital's software business strategy).

  OZIX instrumentation requires either a high resolution clock or a cycle counter for effective and efficient instrumentation.

  OZIX hardware platforms must provide the following:

  – Booting methodology consistent with OZIX on-disk format (e.g., must not require a particular on-disk format)

  – Internationalized console terminal support in "program I/O" mode (e.g., avoiding inappropriate interpretation of characters sent to the console terminal in console program I/O mode.)

  PROVIDER: Decision: Product Marketing based on Hardware and Software Engineering inputs

  Prototypes: Hardware Engineering

  Specifications: Hardware Engineering

  REQUIRED COMPLETION DATE: Decision: March, 1990

  Prototypes: October, 1990

  Specifications: March, 1990

CONTINGENCY PLANS: Delays in hardware specifications, prototypes, and/or product hardware will have an impact on the OZIX V1 schedule.

Without CDROM reader support, OZIX will ship on traditional distribution media, such as magtape. However, this may compromise the schedule (increased testing requirements), security (no read-only distribution media), as well as corporate software business plans.

There is no reasonable contingency which will accommodate a booting methodology that requires a particular on-disk format.

Lack of internationalized console terminal support will result in the shipment of an "almost-internationalized" system. That is, the system will not present a consistent internationalized interface to all system users.

- **DEPENDENCY: 64-bit C Compiler**

  A plan is required for the 64-bit C compiler so that our development plans for migration to a full 64-bit environment can be set.

  PROVIDER: Lu Anne Van de Pas, Dick Wilson

  REQUIRED COMPLETION DATE: 1-Jan-1990

  CONTINGENCY PLANS: Continue with 32-bit development and simulation

- **DEPENDENCY: Internationalized Commands and Utilities**

  There are no plans in place for OSG-OS/CR to do the work for the commands and utilities to utilize compound strings in support of a fully internationalized system.

  PROVIDER: Pete Smith (OS/CR)

  REQUIRED COMPLETION DATE: Committed Strategy: 15-Dec-1989

  Internationalized Commands and Utilities delivered: 15-Jan-1991

  CONTINGENCY PLANS: We will evaluate the possibility of having the internationalization work contracted out, or done by DECwest, with possible V1 schedule impact.

9 Critical International Concerns

OZIX must allow international and multinational customers to interact with the system using individual user choice of local language and cultural conventions.

OZIX meets this requirement in a number of ways:

- All OZIX components facilitate translation by separating program user interfaces and message text from functional code, and by using multilingual messaging and language switching facilities.
- Major OZIX components such as the base system, file system, messaging facility, and part of the terminal services are culturally neutral. That is, there is no assumption made about any particular character set or sets and data may be encoded in any one of a variety of sets.
- OZIX online help and documentation is structured modularly and hierarchically to facilitate translation.
- OZIX terminal subsystem provides support for multiple character sets and languages, and allows for the inclusion of additional locale-specific character set handling without modification of the basic terminal subsystem code or data structures.

- Compound string technology is used throughout OZIX to provide multibyte character handling, to support mixed writing directions, and to provide a data announcement mechanism within a distributed system environment. Through the use of compound string technology, all file names, directory names, user names, etc., may be expressed in native language character sets without loss of writing direction information.

10 Issues Without a Clear Resolution Process

The most critical issue to the project schedule is a clear definition of the FRS hardware platform and its schedule. We are currently assuming the MIPS R4000 on the XMI-2 platform.

11 Budget Requests/Project Tasks and Estimates

The OZIX budget is based on an FY90 headcount of 72 engineers, at $100,000 per engineer. With an anticipated growth in the engineering staff of 10%/year and 10% cost/engineer, the engineering development cost of OZIX V1 is $44M.

Additional anticipated costs beyond headcount include:

- Security consultant (TIS)

  We are negotiating a contract with TIS (a security consultant firm) to help us with the details of NCSC certification. Several of the firm's employees were formerly employed by the NCSC. It is anticipated that their expertise will facilitate the B2 certification of OZIX. We expect this contract to run $250K this fiscal year and $250K next fiscal year.

- Outside contractors

  There are several functional components of OZIX that we are looking for a software contractor to implement. These components are primarily application-level, and none impact the Trusted Computing Base. We anticipate writing contracts totaling $600K over the V1 project cycle to outside contractors.

11.1 Responsibilities

The OZIX team is currently comprised of the following members:

- **Executive and File System**
  - John Penney–Supervisor
  - Jay Bruce–Booting; System Debugger
  - Tom Cockcroft–Transaction Processing
  - Debbie Girdler–File System Project Leader; File Name system
  - Jeff Havens–Nub; General base system
  - Charles Olivier–Base System Project Leader; Process subsystem
  - Dave Orbits–Hardware engineering liason
  - Jim Schirmer–Security
  - Michael Schmitz–Covert channel analysis
  - Chris Saether–Base System and ABA development
  - Pete Stoppani–Media management utilities

  - Jim Walker–Memory management
- **Networks**
  - Marilyn Fries–Supervisor; DECnet/OSI network layer
  - Doug Barlow–DECnet/OSI session; Network security
  - Wim Colgate–Event dispatcher; Entity Database
  - Kevin Dunlap–Network transport interface; ARP
  - Kelly Green–DECnet/OSI Project Leader; DECnet/OSI transport
  - Steve Jenness–TCP; Network Architect
  - Will Lees–NFS
  - Oscar Newkerk–Network Management Project Leader; Node entity, protocol development
  - Miriam Nihart–System Administration agent; Entity support library
  - Charlie Wickham–TCP Project Leader; IP; UDP
- **I/O**
  - Bill Watson–Supervisor
  - Carl Appellof–Terminal Services
  - Terry Carruthers–SCS; CI
  - Sumanta Chatterjee–Mass Storage Project Leader
  - Min-Chih Lu Earl–Miscellaneous I/O Project Leader; LAT; NI
  - Dilip Hardas–Shadowing
  - Paul Saxon–T/MSCP; SCSI
  - Kathy Sestrap–I/O Configuration Management
- **System Administration and RPC**
  - Mark Ozur–Supervisor
  - Mark Ditto–System Administration Project Leader; Installation
  - David Fleskes–Contributing engineer
  - David Lawlor–Management Applications
  - Jim Teague–RPC
  - Thomas Teng–Management Applications; Enabling technologies
- **Performance**
  - Mike Peterson–Supervisor
  - Pete Benoit–Performance Project Leader
  - George Moberly–Performance tools; Performance projects/support
  - Dave Swanson–Performance tools
- **Utilities, RTLs, and Internationalization**
  - Dick Funk–Supervisor

- — Myles Connors–Messaging; Commercialization
- — Claire Cockcroft–Internationalization Program Manager
- — Dennis Doherty–Utilities/commands Project Leader
- — Dan Smith–Libraries/utilities/commands
- — Mike Thomas–Libraries/utilities/commands
- — Krishna Varikooty–Libraries/utilities/commands

- **Testing, Diagnostics, Development Environment**
  - — Dave Snow–Supervisor
  - — Tina Anderson–Development Environment
  - — Richard Brown–ODM Project Leader
  - — Mike Cattolico–Development Environment
  - — Jan D'Addamio–Testing Project Leader
  - — Jeff Lane–OX
  - — Kim Peterson–CASE; Development Environment
  - — Alan Roskey–Testing
  - — Dave Walp–Resource and Fault Management
- **Project Management**
  - — John Gilbert–Development Manager
  - — Benn Schreiber–Project Manager
  - — Patricia Trytten–Project Manager

11.2 Development Resources

11.2.1 Software Requirements

OZIX engineering utilizes many software tools for development. Many of these are standard Digital tools, such as editors, VAX Notes and mail. The project uses the following tools which are either non-standard, or have an impact on hardware resources and/or budget:

- VAX DOCUMENT

  The OZIX project technical documentation is developed using VAX DOCUMENT. VAX DOCUMENT is known for its characteristic of requiring substantial compute power for time-efficient processing.

- OSG C compiler

  OZIX is planning to use the OSG C compiler for its development. This compiler is currently Phase 0, with Phase 1 planned for early next year.

- PROCASE/Atherton

We are currently evaluating PROCASE for use as an integrated program development environment, and Atherton for software configuration management. One of the aspects of these tools that we are looking carefully at is whether these tools can be integrated effectively to provide an enhanced environment for our development. PROCASE is very CPU intensive, and requires a DECstation 3100 for reasonable performance.

- Hardware simulators

  The Sable and R4000 architectural simulators provide an execution environment necessary to test and debug OZIX software components having architectural dependencies.

11.2.2 Hardware Resources

OZIX engineering requires the following hardware resources:

- ULTRIX Timesharing Systems

  Until OZIX is capable of supporting a development environment, we are using ULTRIX for our software development. We estimate that three VAX 8800-class machines are required to support the level of editing, compiling, and applications testing expected during development.

  In addition, we require one VAX 8800 to be running the most current ULTRIX baselevel, for evaluation and testing, and also as an internal field test site for the ULTRIX engineering group. Because these baselevels are internal and have not necessarily had much load testing, we can not consider this machine to provide the same level of availability and reliable service as the timeshare systems running released software.

- VMS Timesharing System

  A large-scale VMS timesharing system is required for the development and storage of project documents, and access to VMS-based software tools such as VAX Notes. Disk storage of about 100,000 disk blocks per engineer is required. Processor requirements are quite significant for support of VAX DOCUMENT and DECwindows-based tools; we estimate 3 VAX 8800 systems and an VAX 8650 provide adequate timesharing response for the OZIX developers, documentation group, and OZIX support groups.

- DECstation 3100 workstations

  The PROCASE tools are extremely CPU intensive, and are only supported on ULTRIX/DECstation 3100 workstations. Each developer actively involved in program development requires a DECstation 3100 workstation.

- ISIS-based Development Systems

  The current development strategy is to use ISIS (R3000 in an CALYPSO/XMI cabinet) as the development platform. We have one CALYPSO/XMI system awaiting an ISIS board set, one ISIS system installed, and two additional ISIS systems in the capital plan for FY90.

11.3 Staffing

Table 9: OZIX Staffing Requirements

FY90	FY91	FY92
72	79	87

11.4 Design Reviews

OZIX design reviews follow a detailed methodology structured to satisfy the requirements of the NCSC B2 certification. The design reviewers consist of the rest of the component project team, engineers utilizing the component, and other subject-knowledgeable engineers throughout the corporation.

Design review discussions and conclusions are captured in VAX Notes conferences for proper project tracking and to satisfy the NCSC B2 certification requirements.

11.5 Project Security

OZIX development is done on timesharing systems located at OSG/DECwest, in Bellevue, Washington. The DECwest Computer Operations Group maintains a reliable computing environment for this project, including:

- Full system backups done on a weekly basis and taken off-site to secure storage
- Nightly incremental backups also retained off-site
- In conjunction with the corporate security organizations, ensures that appropriate precautions are taken against service interruptions due to physical and/or electronic intrusion

Project and design documents are retained over the life of the project in CMS libraries. Design discussions are maintained in VAX Notes conferences, which are retained over the life of the project. This is required for the NCSC certification review, and also provides a history for engineers joining the project.

11.6 Document Retention/Change Control

All OZIX Phase documents are retained in a CMS library on the OZIX timesharing systems. Copies of these documents are available by contacting OZIX Product Management (Beverly Cotton, DECWET::COTTON).

12 Related Documents

- OZIX Vision
- OZIX V1.0 Product Description
- OZIX Support Plan
- OZIX Business Plan
- OZIX Market and Product Requirements
- OZIX Technical Summary
- OZIX Internationalization Plan
- OZIX Master Documentation Plan
- OZIX Test Plan
- OZIX Usability Plan
- OZIX Market Plan
- OZIX Sales Plan
- OZIX Manufacturing Plan

- OZIX Customer Services Plan
- OZIX Application Requirements
- OZIX Functional Specifications

APPENDIX A

PHASE REVIEW PLANNER

Table 10: OZIX Phase Review Planner

Activity	DRI	Target	Actual
PHASE 0/1			
Business Plan	Terry Morris	17-Nov-1989	17-Nov-1989
Engineering Specifications	Benn Schreiber	17-Nov-1989	17-Nov-1989
Engineering Development Plan	Benn Schreiber	17-Nov-1989	17-Nov-1989
Customer Services Plan	LeeAnn Stivers	17-Nov-1989	17-Nov-1989
Internationalization Plan	Mike Oughton	17-Nov-1989	17-Nov-1989
OZIX Technical Summary	Benn Schreiber	17-Nov-1989	17-Nov-1989
Master Documentation Plan	Jim Jackson	17-Nov-1989	17-Nov-1989
OZIX Test Plan	Jan D'Addamio	17-Nov-1989	17-Nov-1989
OZIX Usability Plan	Cheryl Snyder	17-Nov-1989	17-Nov-1989
OZIX Market Plan	Kevin Breunig	17-Nov-1989	17-Nov-1989
OZIX Sales Plan	Joel Berman	17-Nov-1989	17-Nov-1989
OZIX Manufacturing Plan	Rick Seed	17-Nov-1989	17-Nov-1989
OZIX Application Requirements	Cathie Richardson	17-Nov-1989	17-Nov-1989
PHASE 2			
Field Test Plan	Terry Morris	Q3FY91	
Draft Documentation Review	Jim Jackson	Q4FY91	
SPD Draft written	Terry Morris	Q4FY91	
Field Test Documentation	Jim Jackson	Q4FY91	
Field Test Kits Prepared	Benn Schreiber	Q4FY91	
Phase 2 Review	Terry Morris	Q4FY91	
Phase 2 Closure	Terry Morris	Q4FY91	
PHASE 3			
Start Field Test	Benn Schreiber	Q1FY92	
Field Test Update	Benn Schreiber	Q2FY92	
Final Documentation Review	Jim Jackson	Q2FY92	
SPD Submission	Terry Morris	Q2FY92	

Table 10 (Cont.): OZIX Phase Review Planner

Activity	DRI	Target	Actual
SPD Approval	Terry Morris	Q2FY92	
PAC Proposal Submission	Terry Morris	Q2FY92	
PAC Approval	Terry Morris	Q2FY92	
Product Announcement	Terry Morris	Q2FY92	
Start 30-day Minimum Ship	Benn Schreiber	Q2FY92	
Phase 3 Review	Terry Morris	Q2FY92	
Phase 3 Closure	Terry Morris	Q2FY92	
SDC Submission	Benn Schreiber	A2FY92	
PHASE 4			
Phase 4 Review			

OZIX Master Documentation Plan

At Exit from Phase 1

November 1989 - Issue 1.0

Issued by:

OSP/DECwest Publications Manager—Jim Jackson

OZIX Programming Documentation Project Leader—Liz Hunt

OZIX System Administration Documentation Project Leader—Marcia Aguero

OZIX Support Documentation Project Leader—Bill Talcott

Approved by: **Date**

OSP Publications Manager—Paul Hammerstrom

OSP/DECwest Production Manager—Craig Kosak

OZIX Engineering Manager—John Gilbert

digital™

Reviewers:

Roger Heinen, *OSG Group Manager*

OZIX Engineering:

Benn Schreiber, *OZIX Project Manager*
Marilyn Fries, *OZIX Development Supervisor*
Dick Funk, *OZIX Development Supervisor*
Mark Ozur, *OZIX Development Supervisor*
John Penney, *OZIX Development Supervisor*
Mike Peterson, *OZIX Development Supervisor*
Dave Snow, *OZIX Development Supervisor*
Bill Watson, *OZIX Development Supervisor*
Lu Anne Van de Pas, *OSG Compiler Development Supervisor*
Claire Cockcroft, *OZIX Internationalization Program Manager*

Open Software Publications:

Cheryl Snyder, *OSP/DECwest Usability DRI*
Eric Getsinger, *OSP/DECwest Internationalization DRI*
Marie Rountree, *OSP Technical Strategist*
Lee Fogal, *OSP/Spitbrook Publications Manager*
Jim Regan, *OSP/Palo Alto Publications Manager*
Clive Robb, *OSF/Reading Publications Manager*

International Engineering Development:

Martyn Kee, *IED UI Manager*
Mike Oughton, *IED/Reading Product Manager*
Kate Latchem, *IED/Reading UI Engineer*
Colin Walters, *IED/US*

Product Management:

Mike Parker, *Manager OZIX Product Managers*
Kathy Appellof, *OZIX Transaction Services Product Manager*
Terry Morris, *OZIX Product Manager*
Cathie Richardson, *OZIX C & Third Party Applications Product Manager*

Customer Services:

Bill Hilton, *OZIX Customer Services Systems Engineering Manager*
Thomas Siebold, *OZIX CSSE Maintainability Engineer*
LeeAnn Stivers, *OZIX CSSE Product Manager*
Myrna Harrison, *OZIX ESDP Project Leader*

Marketing Communications:

Pete Popieniuck, *MEM Marketing Communication Services Manager*

Manufacturing:

Rose Ramsey, *Software Manufacturing Product Manager*

Digital Equipment Corporation
Confidential and Proprietary

TABLE OF CONTENTS

FIGURES

TABLES

Preface

This document describes the software documentation plans for OZIX[1] as currently defined. The contents of this document are subject to change as the software definition changes.

Individual documentation module plans for each OZIX module are available when completed in the following public directory: decwet::guest1$:[ozix.v1.docplans]. Each documentation module plan includes the writing schedule for the module, production information, and a detailed outline of the module.

Related Documents

- *Internationalization Implementation Plan for OSP/DECwest Documentation*

  This plan is available from Eric Getsinger (decwet::getsinger).

- *OZIX Usability Plan*

  This plan is available from Cheryl Snyder (decwet::snyder).

- *Online Documentation System Functional Description*

  This functional description is available from Rob Shuster (decwet::shuster).

- *Master Documentation Plan for OSG C*

  This plan is available from Liz Hunt (decwet::hunt).

- *OZIX Documentation Packaging Plan*

  This plan will be produced after Phase 1 exit.

Revision History

Date	Issue Number	Description/Summary of Changes
August 1989	0.1	Preliminary internal review copy
September 1989	0.2	Preliminary external review copy
October 1989	0.5	Draft of Phase 1 exit plan
November 1989	1.0	Phase 1 exit plan

[1] OZIX is a code name for a program. For simplicity, the term *OZIX* is used in this document as a product name to refer to the series of products that complete the ULTRIX product family. Products produced by the OZIX program will ultimately be named consistently with ULTRIX products.

1 OZIX Program Overview

This section provides an overview of the OZIX program to deliver a series of high-performance, reliable, open operating systems designed for production environments. These products address the needs of the open systems marketplace and are designed to provide Digital with a leadership role in the production computing market of the 1990s.

1.1 OZIX Program Mission

The mission of the OZIX program is to provide systems that allow Digital to compete in the growing open systems market, particularly where production systems are required. This mission is in keeping with the mission of the Open Software Group (OSG) to supply a complete family of open software products, and with Digital's corporate mission to supply enterprise-wide information systems.

OZIX systems provide solutions for a variety of customer needs in the UNIX marketplace. As *open* systems, they represent a superior implementation of open software standards, incorporating basic system concepts that reflect the models of process management, file processing, and system calls defined in the OSF, X/Open, SVID, POSIX, and OSI standards.

Targeted primarily at customers who require production systems for production environments, OZIX systems incorporate technologies that deliver maximum performance, fault tolerance, and reliability. They are optimized for superior availability and data integrity, providing a solid foundation for transaction processing (TP) and customized applications. In addition, OZIX systems serve as compute and file servers in environments that require enhanced security and reliability.

By using interfaces such as TCP/IP, NFS, OSI, and RPC, OZIX systems are able to interoperate with other open systems. As integral members of multivendor networks, OZIX systems interoperate with UNIX, IBM, VMS, and ULTRIX systems.

By complying with X/Open's XPG3 and OSF's AES level A, applications written for OZIX systems can be easily ported to new hardware platforms. Portability of Digital applications is provided with Application Integration Architecture (AIA) components such as DECwindows client, Compound Document Architecture (CDA), and Concert Multithread Architecture (CMA).

1.2 ULTRIX Familiness

Digital's ULTRIX family spans a broad range of functionality from a low-cost implementation of OSF/1 to the high-functionality implementation of OZIX products. As an integral member of the ULTRIX family, OZIX products share a common family heritage with all ULTRIX products:

- A common set of application programming interfaces (APIs)
- A common set of application development tools, utilities, and commands
- A single network and distributed systems architecture
- A single system and network management architecture
- A common core of documentation

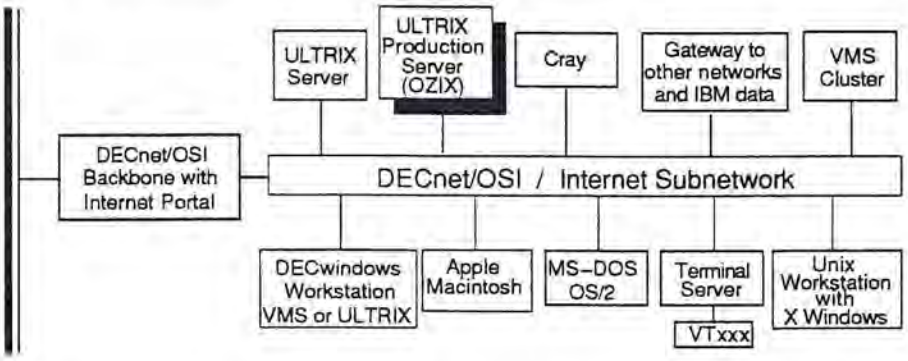
OZIX products are targeted at markets where UNIX (and traditional ULTRIX) systems are not generally used today. They are specifically designed to satisfy needs in the UNIX marketplace where weaknesses exist today. These areas include system security, reliability, availability, file and data integrity, software tools, and documentation. These weaknesses have created a new market, a non-traditional UNIX market, in which the requirements for UNIX systems have grown substantially.

1.3 OZIX Product Description

OZIX is a simple, extensible, hardware-independent operating system. It integrates modern technologies for distributed systems, fault tolerance, and data integrity, while adhering to a high-performance I/O architecture. Its capabilities are implemented over a series of releases and are composed of system elements (such as database and transaction processing software, compilers, tools, and utilities) from other Digital groups, in addition to selected elements from third-party vendors.

The OZIX Version 1 product is a high-performance production system targeted at the open systems market. While transaction processing technologies are built into Version 1, these capabilities are not accessible to the application developer until Version 2. Version 1 is designed to function as a network server in a multivendor, distributed systems network, as shown in Figure 1.

Figure 1: Digital's Computing Environment in the 1990s



Version 1 of the OZIX product serves this network by providing file services (via NFS), compute services (user applications), workstation services (such as workstation system management, booting, and diskless services), and database services. Version 1 also includes integrated system and network management features, license management, and CDROM distribution.

Establishing a successful track record with Version 1 of the OZIX product allows Digital to address more demanding production system requirements with Version 2. The transaction processing technologies built into Version 1 become available to the application developer in Version 2. The goal for this release is to be the highest performance TP system both in terms of transactions per second per MIP, and in terms of dollars per transactions per second. In addition, Version 2 provides world-class distributed system administration via a graphical user interface, while continuing to expand its support for new AIA component architectures.

1.4 OZIX Product Documentation

ULTRIX family products share a common core of information, supplemented by system-specific information for each unique product in the ULTRIX family. The OZIX product information set is structured specifically for online presentation in a hypertext information style. OZIX products will incorporate this presentation style at first revenue ship, although hardcopy documentation is also provided.

The OZIX product information set is tailorable for the needs of different types of users and is designed for easy translation to other languages. In addition, publications tools are being developed to help integrate training modules and documentation from other sources (for example, OSF, third-party vendors, or Digital) into the OZIX product information set.

2 OZIX Documentation Strategy

This section provides an overview of the OZIX documentation strategy, including the overall documentation objective, the relationship to other Open Software Publications (OSP) documentation, the relationship to the Open Software Foundation (OSF), the relationship to third-party vendors, and the relationship to training.

2.1 Documentation Objective

The overall objective for the OSP/DECwest publications group is to provide a state-of-the-art information set for OZIX products that, in itself, provides significant added value to the OZIX products. We intend to accomplish this objective through a strategy based on the following innovative ideas:

- Develop a core of common information for all ULTRIX family products; extend and supplement this core with system-specific information in a consistent manner.
- Tailor the information for online presentation; continue to make documentation available in hardcopy form.
- Structure the information to recognize the needs of different kinds of users:
  - — Use online technology to facilitate quick access to information.
  - — Use progressive disclosure techniques for hardcopy information.
- Optimize the information for easy translation to other languages.
- Meet the requirements for documentation produced in conformance with Digital's AIA information architecture.
- Use the most-sophisticated delivery methods (online information, CDROM distribution, Digital's Demand Printing program).
- Develop publishing tools that allow easy integration of other Digital, OSF, and third-party documentation into the information set.
- Test the usability of the user information set against an accurate, well-defined user model.
- Test the quality of the user information set against a predefined "quality plan" (technical review, editorial review, internationalization review, quality assurance review).
- Coordinate the online information set with ESDP course development.

2.2 Relationship to Other OSP Documentation

The focus of the OZIX documentation produced by OSP/DECwest is on the OZIX added value and uniqueness implemented in the system administration, application programming, and software support environments. The "general user" environment for OZIX is primarily composed of ULTRIX family components implemented by other OSG engineering groups, and therefore, is documented by OSP groups located in Nashua, NH, Palo Alto, CA, and Reading, England, as part of the common core of information shared by all ULTRIX family products. This core of common general information is supplemented by OZIX-specific information.

The goal is to provide documentation for the entire family of open software products that is consistent across all platforms. In order to meet this goal, the documentation for all open software products must be planned, developed, and coordinated across all OSP groups.

2.2.1 OSP Group Responsibilities

OSP/Spitbrook (Nashua, NH) provides documentation for ULTRIX/OSF products that address traditional UNIX markets—products based on a low-cost implementation of OSF/1. OSP/Spitbrook provides documentation for common ULTRIX family components, such as traditional UNIX commands and utilities.

OSP/DECwest (Bellevue, WA) provides documentation for OZIX products that address non-traditional UNIX markets. In these markets, the customers have problems that cannot be satisfactorily addressed by implementations based on the OSF reference code. OZIX products are compliant with the OSF interface specifications where appropriate. By building a platform that is not gated by the OSF reference code, Digital is able to capture market share outside of the traditional UNIX space.

OSP/Palo Alto provides workstation and graphical interface components for both ULTRIX/OSF and OZIX products.

OSP/Reading provides bundled layered products such as the License Management Facility (LMF) and DECprint, and mail services for both ULTRIX/OSF and OZIX products.

In addition, the Open Software Publications Tools Group (OSPTG) supports the publishing tools needs of all OSP publications groups. OSPTG (Bellevue, WA) is a service organization chartered to address the specific needs of groups that produce documentation for an open systems environment. OSPTG is working with the OSF publishing community to arrive at an industry-standard online publishing solution.

2.3 Relationship to the OSF

The Open Software Foundation (OSF) acts as a clearing house and integrator for OSF software and documentation. Integration of OSF documentation into the OZIX information set will occur where applicable to meet our goals. In addition, we will document OSF-compliant functionality and delineate extensions to OSF functionality or behavior.

OSP in general and OSPTG in particular intends to influence, support, and adopt standards for documentation production tools and online documentation access tools specified by the OSF.

2.4 Relationship to Third-Party Vendors

Third-party vendors play a key role in the OZIX program. By offering a rich set of third-party software development tools and utilities, OZIX products reduce development time, thus providing a means for achieving and maintaining a leadership position in the open systems marketplace. The availability of a broad range of third-party application packages also increases Digital's opportunity to sell OZIX products by satisfying the customers' needs for complete *solutions*, not simply *systems*.

OSP has an overall goal to integrate third-party documentation into Digital documentation and into the Software Business Technologies (SBT) Consolidated Distribution (ConDist) and Online Documentation (OLD) programs. Integration is contingent upon acceptance criteria based upon Digital's AIA information architecture. OSPTG is developing publishing tools to help integrate third-party documentation and documentation from other sources (OSF and other Digital groups) into the OZIX product information set.

2.5 Relationship to Training

The DECwest Educational Services Development and Publishing (ESDP) group has expressed interest in using an online approach for OZIX training, complementary to the online focus for documentation. ESDP may provide training in a variety of formats, with varying degrees of integration with the documentation.

OSP/DECwest and ESDP are working together to ensure our deliverables are presented to the user in an integrated and consistent fashion. The training deliverables that integrate closest with documentation are being developed such that they leverage off the documentation, minimizing duplication of effort.

3 OZIX Documentation Production Strategy

This section provides an overview of the production strategy for the OZIX information set. It describes the production tools and techniques used for OZIX documentation to be delivered in online and hardcopy forms. Complete production and packaging details will be included in the *OZIX Documentation Packaging Plan*, to be produced after Phase 1 exit.

3.1 Common Tools and Techniques

A common set of tools is used to create documentation for all forms of delivery. Document formatting is done using VAX DOCUMENT on VMS and *ditroff* on ULTRIX. All illustrations and graphics are created online using VAX RAGS, MOPS, and UtoX.

It is a goal to produce hardcopy documentation from the same source files used to produce online documentation modules. The source files may require conditionalized text that provides introductory and transitional text for hardcopy presentations.

3.2 Online Documentation

OSPTG has added ULTRIX support to the DECwindows Bookreader Version 2.0. The ULTRIX DECwindows Bookreader is an integral part of the corporate online documentation strategy. Making a commitment to online documentation now lets us use the ULTRIX DECwindows Bookreader as a foundation for building the kind of advanced online documentation system required to meet the objective to provide a state-of-the-art information set for OZIX products.

The OSP/DECwest publications group has published a design proposal for a hypertext-style online documentation system that provides a modular presentation of information for OZIX products. This online documentation system addresses the requirements of the publications industry for fast and efficient online documentation, meets the internationalization requirements specified for OZIX products, and meets the OZIX product requirement to be state of the art. The *Online Documentation System Functional Description* is available from Rob Shuster (decwet::shuster).

OSPTG is working closely with the CUP Information Technology Group to ensure that the features described in the *Online Documentation System Functional Description* are implemented in an open systems environment, in line with the corporate strategy for online information creation and delivery. The thrust of this strategy is to integrate DECwindows Bookreader, DECwindows Help, MEMEX hyperinformation services, and Motif with a common authoring tool, such as DECwrite.

It is unknown at this time whether a full implementation of the proposed online documentation system will be available in the OZIX Version 1 timeframe.

As a contingency plan, OZIX information can be delivered online in three formats—ULTRIX DECwindows Bookreader, ULTRIX DECwindows Help, and UNIX-style reference pages—using tools as they exist today. Note, however, that this plan does not allow us to meet the overall objective to provide a state-of-the-art information set for OZIX in the Version 1 timeframe.

- Documentation is processed for the ULTRIX DECwindows Bookreader using both VAX DOCUMENT and *ditroff* (*ditroff* functionality is being provided by OSPTG in 1990). UNIX-style reference pages are provided in ULTRIX DECwindows Bookreader format in addition to the traditional *nroff* format.
- ULTRIX DECwindows Help files are generated with either the SDMLHELP utility or as hand-coded ASCII files.
- All reference pages are coded using the *ditroff* -man macros and formatted using *nroff*.

The online documentation for OZIX is distributed on CDROM in accordance with the SBT ConDist and OLD programs. CDROM production adheres to DEC STD 073.

3.3 Hardcopy Documentation

All hardcopy documents are processed as compound PostScript files. The PostScript files are Electronic Documentation Master Standard (EDMS) compliant and acceptable for use in Digital's Demand Printing program.

Draft and field test documentation is produced on LPS40 laser printers. Final masters are printed on 600 dot-per-inch plain paper typesetters and delivered to the U.S. Area Software Supply Business (SSB) for printing, stocking, and distributing.

4 OZIX Documentation Development Environment Strategy

This section provides an overview of the development environment strategy to be followed by OSP/DECwest to produce the OZIX information set. The goal is to establish a productive, dependable, and sophisticated documentation development environment.

OSP/DECwest intends to use VAX DOCUMENT and *ditroff* for our documentation work, with a long-term goal to use a common authoring tool, such as DECwrite or an SGML-based document processor when it becomes available. We are currently submitting our requirements for a cross-platform authoring system to the CUP Authoring Tools Task Force.

4.1 Documentation Tools Needs

The OZIX information set includes documentation written for ULTRIX, ULTRIX/OSF, and OZIX platforms. In addition, there may be a small amount of sharing at some level with documentation for VMS platforms. Therefore, documentation migration needs include:

- Some sharing with VMS platforms, such as AIA documentation
- The ability to exchange documentation with the OSF
- Interoperability between VAX DOCUMENT and *ditroff*

Table 1 describes the minimum requirements for the OSP/DECwest documentation development environment, and whether or not these components exist today.

Table 1: Minimum Documentation Development Environment Requirements

Component	Description	Exists today	Comments
Text editor with DOCUMENT support	Key bindings, etc., for generating tags, templates, etc.	No on ULTRIX; Yes on VMS	Not high priority. LSE; LSEVE (internal only).
Text editor with ditroff support	Generate macros, etc.	No	Not high priority.
Spell program	Spell checker	Yes	
Online graphics	RAGS, and other non-DEC PostScript file generators	Yes	
Document processors	VAX DOCUMENT and ditroff	Some	See detail below.
Source file control	sccs, make; CMS	Yes[1] on ULTRIX; Yes on VMS	But sccs doesn't have classes.
Indexing capabilities	ULTRIX Doc Tools (UDT) has some; VAX DOCUMENT has it	Yes[1]	But we need to be able to create master indexes across deliverables written with different formatters.
Preview mechanisms	PostScript previewer Help widget sample app'n Bookreader	Yes Yes Yes[1]	 Internal-only. But needs enhancements to meet online documentation system requirements.

[1]The component exists, but does not completely meet our requirements.

4.2 Document Processing Needs

OSP/DECwest has a need to process three kinds of input:

- VAX DOCUMENT source
- *ditroff* source coded in -man and -mu macros (ULTRIX documentation format)
- *ditroff* source coded in -man and -mm macros (OSF format)

The matrix in Table 2 lists the kinds of input formats we need to be able to handle, and the kinds of output we need to be able to generate. Although we already have the tools to do much of the processing, we still lack some of the tools.

We assume that we do not need to handle any other *ditroff* macro packages besides those being used by other OSP groups (-man, -mu) and those being used by the OSF (-man, -mm). For example, we do not currently anticipate a need to handle -me macros or third-party macro packages.

Table 2: Document Processing Input/Output Requirements

	Input		
Output	DOCUMENT source	ditroff (ULTRIX) man, mu	ditroff (OSF) man, mm
PostScript	Have	Have	Have
Bookreader files	Have	Need	Need; unless create mm-to-mu converter
ditroff source man, mm	Need; for submit to OSF	Need; for submit to OSF	-
ditroff source man, mu	Not needed	-	Need
DOCUMENT source	-	Need; for I18N	Not needed; use mm-to-mu, then mu-to-DOCUMENT

Note that we do not currently have any dependencies on DECwrite for document processing input/output requirements. This could change, however, once the corporate strategy for online information creation and delivery is finalized. If DECwrite is selected as the common authoring tool, we may need to process DECwrite source in addition to DOCUMENT and *ditroff* source.

4.3 Summary of Requirements

The following summarizes the requirements for the OZIX documentation development environment, ordered from high to low priority:

1. Enhanced Bookreader functionality to meet hypertext-style online documentation system requirements.
2. A way to create master indexes across deliverables written with different formatters.
3. A way to process DOCUMENT files from ULTRIX. This can be satisfied by porting DOCUMENT to ULTRIX, or by creating a way to submit files from ULTRIX to be processed on a VMS system (maybe employing the ULTRIX Connection (UCX) product).
4. A *ditroff* (man, mu)-to-DOCUMENT converter. (For internationalization: multinational font capabilities, and translation groups are currently focused around DOCUMENT.)
5. Create Bookreader files from *ditroff* (man, mu) source. (To put ULTRIX books online.)
6. A *ditroff* (man, mm)-to-*ditroff* (man, mu) converter. (To integrate files from the OSF.)
7. Create Bookreader files from *ditroff* (man, mm). This can be satisfied with an mm-to-mu converter [6], plus a way to create Bookreader files from mu source [5].
8. Additional features in *sccs*, such as classes.
9. A DOCUMENT-to-*ditroff* (man, mm) converter. (For submissions to the OSF.)
10. A *ditroff* (man, mu)-to-*ditroff* (man, mm) converter. (For submissions to the OSF.)
11. A way to generate DOCUMENT tags from an ULTRIX text editor.
12. A way to generate *ditroff* coding macros from a text editor.

5 OZIX Internationalization Strategy

This section provides an overview of the internationalization strategy for the OZIX information set, including an internationalization overview, internationalization goals for documentation, and a description of the functions of the International Engineering Development (IED) user interface engineer and the OSP/DECwest translation coordinator.

5.1 Internationalization Overview

An international product is one that meets the requirements of international and multinational customers to allow them to interact with the system using local language and cultural conventions. An international product can be easily localized and is designed for the international market.

OZIX products are designed to be sold in international markets and therefore may be described as language-neutral, culturally unbiased software products. Internationalization is built directly into the products, including the capability to accommodate worldwide character sets for textual representations. Such features ensure that customers and third-party software developers can create applications tailored for specific cultures and local languages.

OZIX products are designed for simultaneous worldwide release. A single OZIX system can support multiple localized user environments, and contains components that provide multiple language-specific interfaces. Text is isolated to facilitate quick and easy translation, and documentation and online help are provided in a form that can be presented in the language of the user, using local cultural conventions.

The *Internationalization Implementation Plan for OSP/DECwest Documentation* describes in detail the OSP/DECwest plan to support Digital's goal of producing international products. This plan is available from Eric Getsinger (decwet::getsinger), the OSP/DECwest Internationalization DRI.

5.2 Internationalization Goals

Internationalization goals for the OZIX information set are to:

- Optimize the information set for easy translation to other languages.
- Structure the information set to recognize the needs of different kinds of users.
- Design and write the information set in modular form to allow for partial translation.
- Ensure the development of online documentation tools that support the creation and display of European, Asian, and Semitic languages and appropriate writing directions.
- Support multiple-language versions of the online information set concurrently, with the capability to select primary, secondary, and default language preferences.

The OZIX information set is designed to optimize translation to other languages by adhering to the guidelines described in the *Producing International Products Reference Set*, or PIP, and the *Planning for Translation* document. Translation is facilitated by following the concepts of core text and modular design, as described in the PIP.

It is a goal to maximize the commonality across the ULTRIX family of open software products. This common information forms a set of core text that remains the same across the family of products, minimizing redundancy and thereby, facilitating translation.

OZIX documentation is written in a modular style. Information is separated into discrete modules by audience, and then by task. This modularity provides maximum translation flexibility, allowing Digital Local Engineering Groups (LEGs) to translate selected information only and to structure that information as required to meet the needs of the local language and culture—without rewriting the original user information.

OSP/DECwest and OSPTG intend to work closely with the CUP Information Technology Group (CUP/ITG) to ensure that the corporate strategy for online information creation and delivery meets the internationalization requirements specified by IED for all OZIX products. Online documentation tools must support the creation and display of European, Asian, and Semitic languages and associated mixed writing directions. They must also support concurrent multiple-language versions of the online information set, with the capability to select primary, secondary, and default language preferences. In addition, all tools must be available to IED and the Digital LEGs.

5.3 IED User Information Engineer

IED assigns a user information (UI) engineer to a project before the close of Phase 1. Timely assignment of this individual ensures that the UI engineer has an opportunity to review all user information, including documentation, screen interfaces, and course materials. Kate Latchem is the UI engineer in IED/Reading responsible for the local-language versions of OZIX documentation in Europe and in the General International Area (GIA) countries, excluding Asia.

The UI engineer manages the transfer of user information into the translation process. This work involves participating in the efforts of the OZIX documentation team, helping the Digital LEGs produce translation plans, receiving source text files for translation, preparing files and information for use in the translation effort, and helping to plan for the production of translated documentation. The UI engineer uses the product under development, and may also provide training to the translators in the use of the product, documentation and graphics tools, and so on.

The UI engineer works closely with the OSP/DECwest translation coordinator to ensure the successful transfer of information into the translation process. The UI engineer provides feedback to the translation coordinator on the documentation plans and designs, the documentation source text files, and the document drafts (both text and graphics), pointing out potential problems to European and GIA customers.

5.4 GIA Translation Group Representatives

In addition to the UI Engineer assigned by IED, OSP/DECwest expects to work directly with GIA translation group representatives in Asia. The GIA translation group representatives have similar responsibilities to those of the UI engineer, managing the transfer of user information into the translation process, and working closely with the OSP/DECwest translation coordinator to ensure the successful transfer of this information.

Note that the GIA translation group representatives for Asia are not yet assigned.

5.5 OSP/DECwest Translation Coordinator

OSP/DECwest assigns a translation coordinator to a project before the close of Phase 1. Eric Getsinger is the OSP/DECwest translation coordinator for OZIX documentation.

The translation coordinator reviews documentation plans and document drafts for compliance with internationalization policy and standards, reviews and distributes comments on the documentation plans and document drafts received from the UI engineer and GIA translation group representatives, and ensures that translatable source files are made available to the UI engineer and GIA translation

group representatives, as appropriate. In addition, the translation coordinator keeps the UI engineer and GIA translation group representatives informed of the status of the documentation, changes to plans, documentation schedules, and so on.

OZIX products are particularly targeted to the Asian marketplace. In fact, initial translation requirements are concentrated in Japan, Taiwan, and Korea. For this reason, it is critical that the OSP/DECwest translation coordinator begin working now through the UI engineer or directly with GIA translation group representatives in Asia to ensure the success of the translation effort.

6 OZIX Usability Strategy

This section describes the usability strategy for the OZIX information set, including a usability overview and the usability engineering plans for the information set.

6.1 Usability Overview

Usability engineering is the design, testing, and evaluation of the user, or human interfaces of a computer system, in partnership with the intended users. Usability engineering for the OZIX product enables the DECwest documentation group to observe, poll, and test users at customer sites and in the usability lab.

For the documentation interface, because documentation is primarily focused on online delivery, usability design and testing for online documentation focuses on evaluating online interfaces and the usability of online access methods. In addition, usability tests whether levels of information documented are appropriate for online task and reference information.

Evaluation of usability relies on finding and using the most accurate, well-defined user pool possible. Methods of evaluation include both contextual and quantitative:

- Contextual evaluation involves gathering information through visiting, observing, and polling users in their own environment.
- Quantitative evaluation involves using more structured settings, such as the usability lab, and more precise methods and metrics to measure user response to specific, carefully defined tasks.

Because OZIX is an international product, we are concerned about international usability and are working with international groups within Digital for input from international user groups to determine if the criteria for usability in the U.S. is acceptable to international users in Asian and European markets.

The *OZIX Usability Plan* defines the overall user model and describes in detail the usability plan for the entire OZIX project. This plan is available from Cheryl Snyder (decwet::snyder), the OSP/DECwest Usability DRI.

6.2 Usability Plans

Usability engineering plans for the OZIX information set are to test the design and implementation of system administration documentation in particular, in keeping with usability plans for engineering. Interfaces are: online documentation for the configuration application, the DECwindows Storage Manager, and online and hardcopy documentation for the installation and startup process. This involves:

1. Testing with internal users of system administration interfaces on early prototype testing.
2. Facilitating observation and polling of users at appropriate customer sites by writers.

3. Evaluating and transmitting results back to documentation, to CUP/ITG, and to product management as market and product requirements.
4. Testing the interfaces before and during field test, as research results are incorporated into prototypical and product interfaces:
   - — Testing online system administration documentation
   - — Testing hardcopy installation documentation

The following questions comprise the basis for the test plan for documentation:

- Do navigation and presentation methods of the online documentation system work for various user groups?
- Are user tasks grouped appropriately for ease of use by target users?
- What level of information is necessary in order to perform a task adequately?
- Is the level of documentation appropriate for the access point?
- How long does it take the administrator to find the location in the documentation?
- Is hardcopy installation documentation well-organized?
- Where do users have problems?
- Is terminology familiar to users?
- How does the interface work for international users when compared to the results of testing with American users?

7 OZIX Review Strategy

This section provides an overview of the review strategy for the OZIX information set, including technical review, editorial review, internationalization review, and quality assurance review.

7.1 Technical Review

The OSP/DECwest writing group relies on timely and thorough technical reviews in order to ensure high-quality documentation. The review strategy for the OZIX information set is to include both internal (to DECwest) and external reviews of the information.

At a minimum, the information set will undergo formal first draft and field test reviews. In most cases, information to be reviewed is available to reviewers online and can be reviewed in an online format. For graphical interfaces, in addition to reading the text, reviewers need to invoke the appropriate applications and "click" on input fields and HELP buttons to review the information. Feedback may be provided via an online text file or via hardcopy markup, as appropriate.

DECwest software engineers review the information at all stages of the project for technical accuracy and completeness. DECwest Customer Services personnel review drafts of the support information through each stage of development.

Non-DECwest software engineers, writers, Customer Services personnel, ESDP personnel, and other interested parties within Digital, in addition to all field test sites, review field test drafts of the information. Feedback on field test drafts may be provided via a notes conference, online text files, hardcopy markup, or other means, as appropriate to the reviewer.

7.2 Editorial Review

Thorough editorial reviews are also required to ensure high-quality documentation. To meet this requirement, the OSP/DECwest production group performs two editing functions: one for text and one for graphics. Text editors check for spelling, grammar, and adherence to the Corporate Documentation Handbook style guidelines. Graphic editors check for illustration placement, consistency of nomenclature between text and graphics, and correct online and hardcopy formatting.

In addition, the writing group supervisors and project leaders review the information set for overall consistency and style.

OSP is developing a requirements document for importing documentation into OSP information sets. Documentation that is to be imported from third parties will be reviewed against these requirements and against the *Corporate Documentation Handbook* style guidelines as part of the overall editorial and quality assurance reviews.

7.3 Internationalization Review

The OZIX information set is written with the international market in mind. OSP/DECwest style guidelines call for modular organization, clear writing, and an extensive glossary. All modules are prepared in conformance with the PIP and, when a document is scheduled for translation, a standard translation pack (STP) is created and submitted with the online and paper copies.

The translation coordinator reviews documentation plans and document drafts for compliance with internationalization policy and standards. In addition, the translation coordinator reviews and distributes comments on the documentation plans, documentation source text files, and document drafts received from the UI engineer or GIA translation group representative, who has an opportunity to review all user information.

7.4 Quality Assurance Review

Quality assurance is critical to the successful completion and delivery of a state-of-the-art information set for OZIX products. Every effort will be made to ensure that quality is integral to the documentation development, production, and distribution.

The quality assurance requirements for hardcopy and online manuals are similar:

- For hardcopy manuals, a quick visual inspection enables a tester to make sure that all pages, figures, and tables are in the book, that page numbers and running headings are accurate, and that an index and a table of contents are included. Verifying the accuracy of code examples requires the greatest amount of time.
- Online manuals require more vigorous, and thus time-consuming, testing of ALL document attributes. Verifying the accuracy of code examples requires the same high level of effort as it does for hardcopy manuals.

8 OZIX Information Set Overview

This section provides an overview of the OZIX information set. It addresses topics common to all areas of the information set, including the purpose and scope of the information set, common objectives, and sources of information, followed by a summary of the information set.

8.1 Purpose

The purpose of the OZIX information set is to provide users with complete, concise, and accurate information that is easily accessible and easy to comprehend. This enables users to accomplish their tasks more efficiently, which in turn, reduces costs and increases revenues. As a result, OZIX documentation can be used as a way of achieving competitive advantage.

8.2 Scope

The OZIX information set provides all conceptual, reference, and procedural information needed by general users, system administrators, programmers, and software support personnel. The information set ensures complete information is available for OZIX, yet maximizes the commonality with other ULTRIX family platforms.

8.3 Objectives

The following objectives are common to all areas of the OZIX information set, ordered from high to low priority:

1. Add value to the OZIX product by providing a quality, state-of-the-art information set.
2. Maximize the commonality with ULTRIX family products and provide a consistent information set across all levels of related documentation.
3. Tailor the information for online presentation, allowing customers and support personnel to access all information online.
4. Optimize the information set for easy translation to other languages.
5. Structure the information to recognize the needs of different kinds of users.
6. Design and write the information set in modular form, using progressive disclosure techniques for hardcopy information.
7. Meet the requirements for documentation produced in conformance with Digital's AIA information architecture.
8. Meet production requirements, including editing, illustration development, and final production requirements.
9. Use the most-sophisticated delivery methods (online information, CDROM distribution, Digital's Demand Printing program).
10. Maximize the usability of the information set by customers and support personnel through usability design, testing, and evaluation.
11. Provide Customer Services personnel with easy access to Digital-internal and customer-visible information, as determined by license management.
12. Provide documentation to Customer Services personnel that enables quick, efficient, and accurate customer support.
13. Support changes to the documentation that originate from either customers or Customer Services personnel through a streamlined method, allowing changes to be provided to the originator and all other users in a minimum period of time.
14. Provide online documentation that provides linking capabilities for computer-based instruction (CBI) applications and minimizes duplication between training courses and documentation.

8.4 Sources of Information

Information for the OZIX information set comes from the following sources:

- Discussions with OZIX engineers
- Engineering specifications, including technical overviews, functional specifications, interface specifications, and detailed design specifications
- Project-related information, including phase review documentation, slide presentations, meeting summaries and minutes, white papers, trip reports, and miscellaneous documents and articles
- Hands-on testing
- ULTRIX family documentation
- VMS documentation
- Open Software Foundation
- Application Portability Architecture (APA) group
- Software Development Technologies (SDT) group
- Networks and Communications (NaC) group
- OSP personnel
- Usability personnel
- Customer Services personnel
- Training personnel

8.5 OZIX Information Set Summary

Table 3 summarizes the primary topics to be included in the OZIX information set.

Table 3: OZIX Information Set

General User Environment
OZIX Reader's Guide
OZIX Release Notes
OZIX Master Glossary
OZIX Master Index
Common user information
DECwindows/Motif user information
OZIX-specific user information
Common man pages
OZIX-specific man pages

Table 3 (Cont.): OZIX Information Set

System Administration Environment
OZIX System Administration Overview
System Programming
Network Administration
Account Administration
Security Administration
Trusted Facility Manual
Security Facility Users' Guide
Routine Operations
Man pages
Management Control Language (MCL) reference material
Application Programming Environment
Introduction to the OZIX Application Programming Environment
Building Applications
Writing Applications for OZIX
Writing Portable Applications
Writing Application Interfaces
Writing Distributed Applications
Writing Multithread Applications
Writing Secure Applications
Writing International Applications
Writing Graphics Applications
Writing CDA Applications
Using INGRES, SQL
Porting Applications to OZIX
Improving Application Performance
Other reference material
Using C
Software Support Environment
OZIX Installation
OZIX System Messages
OZIX Troubleshooting Tree
OZIX Troubleshooting Guide
OZIX Internals

Section 9, Section 10, Section 11, and Section 12 describe in detail the general information, system administration, application programming, and software support information sets, respectively.

9 OZIX General Information Set

This section describes the general information set that is provided for all OZIX users. Because the focus of the OZIX documentation produced by OSP/DECwest is on the OZIX added value and uniqueness implemented in the system administration, application programming, and software support environments, most "general user" information is documented by other OSP groups as part of the common core of information shared by all ULTRIX family products. This core of common user information is supplemented by OZIX-specific user information.

Table 4 lists the topics in the OZIX general information set, together with their estimated page counts.

Table 4: OZIX General Information Set

Topic	Page Count[1]	OSP/DECwest Writers	Other Writers
OZIX Reader's Guide	10	tbd	
OZIX Release Notes	50	tbd	
OZIX Master Glossary	100	tbd	
OZIX Master Index	400	tbd	
Common user information	tbd		OSP/Spitbrook
DECwindows/Motif user information	tbd		OSP/Palo Alto
OZIX-specific user information	tbd	tbd	
Common man pages	tbd		OSP/Spitbrook
OZIX-specific man pages	tbd	tbd	

[1]The page-count definition may require redefinition to present a more accurate appraisal of the online documentation development. These page-count estimates are based upon traditional, hardcopy algorithms.

The general information set for OZIX is not fully addressed in this version of the documentation plan. A later version will describe in detail the common general user information that must be planned, developed, and coordinated across all OSP groups, in addition to the OZIX-specific information required to supplement this core of common user information.

10 OZIX System Administration Information Set

This section describes the information set that is provided for OZIX system administration. It includes an overview of the OZIX system administration environment, an audience definition, the added value that the system administration information set specifically provides, the online and hardcopy documentation strategy, and a summary of the system administration information set.

Hereafter in this section, the term "system administration" refers to both system and network administration.

10.1 System Administration Environment Overview

The system administration component of OZIX products is designed to provide system administration services in a multivendor, production environment. The OZIX system administration component is compatible with Digital's Entity Management Architecture (EMA).

It is a goal of the OZIX program to simplify the overall task of system administration as much as possible. By making system administration easier, OZIX products allow customers to reduce the size and expertise of their operational staff, thus reducing the overall cost of system administration.

There are four essential aspects of OZIX system administration that affect how we document system administration tasks. These aspects are:

- Centralized, distributed system administration

  This means that, from a central location, system administration personnel will use OZIX to manage all nodes in the configuration.

- Various user interfaces to system adminstration tasks

  There are four types of user interfaces that OZIX provides for system administration. They are:

  — The Management Control Language (MCL) command-line interface

  As a way to more fully integrate system and network administration, including the administration of DECnet-OZIX, OZIX provides the Management Control Language (MCL). MCL is a set of commands and qualifiers that consists of the DECnet Phase V Network Control Language (NCL) (used for managing DECnet-OZIX), as well as additional commands for all other OZIX system administration tasks. These additional commands adhere to the look-and-feel of Phase V NCL. MCL allows a system administrator to manage any manageable object registered with the OZIX management backplane.

  — Graphical interfaces

  Graphical interfaces are provided for system administration tasks, based on OSF/Motif style guidelines. For OZIX Version 1, there are two types of graphical interfaces being provided. The first type is an easy-to-use point-and-click browser/editor for managing any object registered with the management backplane. This interface is an OSF/Motif-based counterpart to the command-line based MCL.

  The second type is an OSF/Motif-based file system/IO resource manager. This interface is a specialized management application that brings out the superior file system/IO architecture design and implementation within OZIX. The interface allows a mix of NFS, ABA, and IO resource management. Other graphical, system administration interfaces as robust as the file system/IO resource manager will be provided for OZIX Version 2.

  — A complete, OSF-compliant command-line interface

  In addition to MCL, there is another set of commands and utilities, along with their options and arguments, that system administrators can use to manage the configuration. This set of commands and utilities comes primarily from the Open Software Foundation, although new commands and utilities are created for OZIX-specific functions. In some cases, existing ULTRIX utilities are ported to OZIX products.

  — B2-level security certification

  One of the B2 security requirements for OZIX products is that system administration functions be engineered according to defined administrative roles. Examples of administrative roles are operators, account administrators, and security administrators. Developing the system administration software according to these roles means that specific types of administrators (such as operators) can only perform a certain subset of tasks; that is, only a subset of system administration commands and utilities is available to them.

10.2 Audience

The audience for the OZIX system administration information set encompasses six groups (called *administrative roles*), as defined by the National Computer Security Center (NCSC). A user might perform from one to five of these roles, depending on how the customer site is set up. These roles and their definitions are:

- Account Administrator—An administrative role or user assigned to maintain accounting files, tools, user accounts, and system statistics.
- Auditor—An authorized individual, or role, with administrative duties, which include selecting the events to be audited on the system, setting up the audit flags which enable the recording of those events, and analyzing the trail of audit events.
- Operator—An administrative role or user assigned to perform routine maintenance operations and to respond to routine user requests.
- Security Administrator—An administrative role or user responsible for the security of the operating system and having the authority to enforce the security safeguards on all others who have access to the operating system (with the possible exception of the auditor).
- System Programmer—An administrative role or user responsible for the trusted system distribution, configuration, installation, and non-routine maintenance.
- Network Administrator—An administrative role or user responsible for setting up and maintaining communications among nodes in the OZIX configuration.

There are five main reasons why the system administration information set is targeted to the above administrative roles:

- OZIX commands and utilities are being designed according to these audience groups. (This is a B2 requirement.) For example, only an account adminstrator is allowed to execute the commands for assigning user accounts. Given this situation, documenting system administration tasks according to these groups is the most reasonable approach.
- Customers are likely to need information for performing some of the more routine, simplified tasks, in an effort to reduce their overall system administration costs. Thus, it is necessary to provide information specifically for those responsible for performing routine operations.
- Commerical environments, especially traditional management information services (MIS) shops, are already accustomed to having at least two of these audience groups manage their systems—system programmers and operators. Thus, a customer's existing environment might already be set up this way to manage an OZIX system.
- Because of enhanced security concerns in commercial industries such as banking, having operators that routinely perform specific, simple tasks may be required. This is to prevent a user from intentionally or unintentionally altering something in the system. Typical examples of functions that users may not be allowed to do (and thus require an operator) are deleting a job or a process, and mounting a tape.
- For the *Trusted Facility Manual*, one of two books required for B2 certification, the National Computer Security Center (NCSC) recommends that the security information be documented according to the above adminsitrative roles, rather than according to straight functionality.

Regarding the level of technical expertise required by an administrator, the information set assumes that administrators have no previous experience managing a UNIX-based system.

10.3 Added Value

The OZIX system administration information set adds value in the following ways:

- The information set describes how to manage an OZIX server and its clients from a centralized location.
- The information set supports the OZIX engineering goal of fully integrating system and network administration tasks. Thus, the information focuses on describing how to manage both the system and the network (including DECnet-OZIX) from within the same user interface.
- For any graphical interfaces that OZIX provides for system administration, the system administration information set is highly integrated into those system administration interfaces. For example, system administration information is immediately accessible from any window of a graphical system administration interface (this is known as context-sensitive help).
- Information provided for command-line interfaces is described in a clear, concise, and straightforward manner, providing ample examples and illustrations where appropriate. In this case, additional information beyond the traditional OSF man pages is provided, allowing the man pages to function primarily as reference material.
- The system administration information set is highly task-oriented, allowing system administration personnel to manage an OZIX configuration in a simple, staightforward manner. The information set enables some system administration functions to be performed by less technically sophisticated personnel, thus reducing the overall cost of system administration.
- The system administration information set enables the customer to maintain a B2 security level, which is particulary important for production environments. It also allows the administration team to maintain strict security controls by providing security administration information to all types of OZIX administrators.

10.4 Strategy

The strategy for the OZIX system administration information set encompasses several areas: online presentation strategy, hardcopy strategy, and the relationship to other documentation. The following subsections describe these areas.

10.4.1 Online Presentation Strategy

It is assumed that the entire OZIX system administration information set is available online. Information that must be provided in hardcopy form, such as OZIX installation information, is also available online.

The system administration information is closely integrated into any graphical interfaces that are available for system administration. For example, if an operator is using an OSF/Motif interface to perform a backup, the operator can "click" on any input field of an interface window to obtain information. More general help for the system administrator is also available within a system administration application, by clicking on the window's HELP button. Presenting information in a context-sensitive fashion is the primary way that system administration information is implemented online, wherever possible.

For any system administration task, the administrator is always able to traverse a link to obtain related concepts or reference information.

Information is categorized according to major functions, which correspond to administrative roles. Within each function, information is divided into several topics or modules.

10.4.2 Hardcopy Strategy

It is a goal to minimize hardcopy output of system administration information as much as possible. Types of information that are likely to be available to customers in hardcopy form are OZIX installation information, and information on handling system problems (troubleshooting).

As with the rest of the OZIX information set, hardcopy documentation will be produced from the same source files used for online information. Producing the system administration information set in hardcopy form will provide a unique challenge because so much of the online information is implemented in a context-sensitive way.

Like the online information, hardcopy information is categorized according to administrative functions. It is likely that information for each of these major functions will represent a separate book.

10.4.3 Relationship to Other Documentation

The OZIX system administration information set has the following relationships to other documentation:

- Parts of the information set will be used by field support personnel in some instances, in which case those parts function as a supplement to the support information set.
- Most of the man pages that are part of the system administration information set will come from the Open Software Foundation. We will add, delete, and modify these man pages where appropriate to accommodate unique features of OZIX.
- We will use existing ULTRIX source files where appropriate.

10.5 System Administration Information Set Summary

The OZIX system administration information set consists of any information that a system administrator (including a network administrator) needs to perform system administration tasks. At the highest level, most of the information is categorized according to adminstrative functions (listed and defined in Section 10.2.) Table 5 lists the topics in the OZIX system administration information set, together with their estimated page counts.

Table 5: OZIX System Administration Information Set

Topic	Page Count[1]	OSP/DECwest Writers
OZIX System Administration Overview	100	tbd
System Programming	250	tbd
Network Administration	300	Debbie Walkowski, tbd
Account Administration	100	tbd
Security Administration	300	tbd
Trusted Facility Manual		
Security Facility Users' Guide		
Routine Operations	250	Marcia Aguero
Man pages	100	Marcia Aguero, Debbie Walkowski
Management Control Language (MCL) reference material	200	tbd

[1]The page-count definition may require redefinition to present a more accurate appraisal of the online documentation development. These page-count estimates are based upon traditional, hardcopy algorithms.

The information for each administrative role consists of modules, or topics, corresponding to specific tasks performed by that role. Each administrative role represents an item on the top-most system administration menu within the online documentation system.

The following list shows the highest-level information categories and the tasks to be documented for them. Note that three of the categories do not directly correspond to a particular administrative role; the reasons for this are explained within the list.

- *OZIX System Administration Overview*

  Audience: All types of administrators

  Because most of the information set corresponds to administrative roles, we need a set of information that provides a high-level overview of OZIX system administration—what tasks are required, who should perform them, when the tasks must be performed, and so on. This information is addressed from a conceptual perspective only; task-oriented information, along with conceptual information pertinent to each task, is documented in other topics such as *System Programming* and *Network Administration.*

- *System Programming*

  Audience: System programmers

  This topic consists of any information that the system programmer needs to perform his or her job. Typically, the system programmer handles any non-routine and technically sophisticated tasks outside of network management (for example, software installation and creating device special files).

- *Network Administration*

  Audience: Network administrators

  This topic includes any information related to managing the network. The two main components documented in this topic are TCP/IP and DECnet-OZIX. Management of other services, such as Bind, Kerberos, and NFS is also described.

  The OZIX system administration information provides DECnet information in such as way as to enable smooth integration between DECnet administration and other OZIX network administration tasks.

  Any descriptions of OZIX implementations of networking protocols (such as TCP, IP, ICMP, UDP, and ARP) are documented as part of the OZIX operating system internals information, and not as part of the system administration information set.

- *Account Administration*

  Audience: Account administators

  The account administration topic describes how to manage two specific areas of the system—user accounts and system accounting.

- *Security Administration*

  Audience: All types of administrators, Programmers

  The security administration information falls into two categories:

  — The *Trusted Facility Manual* (B2 requirement)

Required for B2 certification, the *Trusted Facility Manual (TFM)* contains all information needed to implement, maintain, and monitor B2-level security on an OZIX system. The information is directed toward all types of administrators, namely system programmers, network administrators, auditors, security administrators, account administrators, and operators. The *TFM* contains a section for each type of administrator, describing his or her responsibilities with respect to system security. The *TFM* is especially important to the security administrator.

It is assumed that implementing the *TFM* online satisfies the B2 requirement for this document.

— *Security Facility Users' Guide* (B2 requirement)

Audience: System Administrators, Programmers

The primary purpose of the *Security Facility Users' Guide (SFUG)* is to document any security information that a non-administrative user might need, such as how to set file protection and choose a secure password. This information also functions as prerequisite reading for the system administration staff.

This information is required for B2 certification.

- *Routine Operations*

  Audience: Operators

  This information is directed to those administrators assigned to perform routine operations on the OZIX system. Examples of topics documented in this part of the information set are: printer setup and management, job control, and backup and restore.

  Many of the tasks normally performed by a general user, such as deleting a print job, are performed by the operator instead, at the user's request. This is a necessary effect of B2 security certification and is in line with traditional commerical environments.

- *Man pages*

  Audience: All types of administrators

  The man pages related to system administration will come from the Open Software Foundation. We plan to modify those man pages wherever appropriate due to OZIX-specific requirements. The man pages serve as the documentation for any UNIX-style commands.

- *Management Control Language (MCL) reference material*

  This information contains the syntax and descriptions of the MCL commands that are used to manage any objects registered with the management backplane (including DECnet objects).

11 OZIX Application Programming Information Set

This section describes the information set that is provided for OZIX application programming. It includes an overview of the OZIX application programming environment, an audience definition, the added value that the programming information set specifically provides, the online and hardcopy documentation strategy, and a summary of the application programming information set.

11.1 Application Programming Environment Overview

OZIX products are designed to provide an application programming environment that increases programmer productivity and decreases application time-to-market. The environment provides a full set of standard programming interfaces, to further application portability and interoperability. OZIX provides a robust operating environment for applications, with superior price/performance among open systems platforms.

The programming environment provided by the base OZIX system can be summarized as follows:

- Standard application development utilities and commands, for example, *ld, dbx, prof, sccs, make, lex, grep*, and *awk*
- Added-value application development commands and utilities, including support for enhanced message catalogs, and support for building and loading subsystems
- Standard application programming interfaces (APIs) as specified by the OSF AES, X/Open XPG3, POSIX 1003.1 and the ANSI C standard:

  General purpose routines
  Math routines
  Curses screen handling routines
  Primitive database management routines
  Socket/XTI interface routines
  OSF/Motif support routines
  X11 Toolkit and widget support routines
  GKS and PHIGS routines

- Added-value routines:

  DECnet routines
  DECwindows Toolkit routines
  Compound Document Architecture DDIF routines (support, viewer)
  RPC routines
  Security support routines
  Compound string support routines
  Bundled application interfaces to INGRES and for SQL support (third-party)
  Concert Multithread Architecture (CMA) routines
  Online Diagnostic Monitor (ODM) routines
  Logging and error recovery routines
  DECprint routines

- Bundled programming languages (ANSI C)

In addition to the environment provided by the base system, other features are provided by Digital layered products and by third-party products (for example, COBOL, FORTRAN, and CASE tools).

11.2 Audience

The primary audience for the OZIX application programming information set is programmers developing commercial applications for the open systems production market. A secondary audience is programmers developing technical applications, who are using OZIX because it is better than the other open systems platforms.

As with traditional UNIX systems, programmers on OZIX require standard application programming interfaces (APIs) and utilities. Beyond standard functionality, however, they also require leadership performance, increased application and data availability, stringent security, and increased maintainability and extensibility.

Commercial programmers

Commercial programmers can be divided into two categories: those who deal only with the tools layered on top of the system (CASE environment, database tools, forms managers, etc.), and those who also interact with the operating system features. The documentation on the development environment is applicable to both types. The rest of the OZIX programming information set addresses the second category of programmers. Programmers who deal mostly with the layered tools are addressed by the tools documentation.

The commercial programming audience that interacts with the operating system features is characterized as follows:

- Applications—Commercial programmers on OZIX Version 1 are developing applications such as telecommunications systems and non-mission-critical accounting packages.
- Software experience—They are experienced in application development, and are most familiar with MVS and its related components (such as JCL, TSO, VM, and CMS). They may have used some flavor of the UNIX operating system. Their primary programming language is COBOL, with some use of C in the telecommunications field, and some use of FORTRAN for financial modeling.
- Customer category—Commercial programmers on OZIX work for end-user companies and for Digital third-party software vendors. Some are Digital engineers developing commercial layered products.
- New versus old code—The majority of our commercial customers for OZIX Version 1 will be writing new applications for this platform, rather than porting over old code.
- Standards—OZIX programmers are very concerned with adherence to standards. A small percentage of the time they may take advantage of added-value features, but they isolate nonstandard code into separate modules.
- Interoperability—OZIX commercial applications need to interoperate with the following environments, from most prevalent to least:

  MVS
  Other UNIX environments (System V, XENIX, and SunOS)
  VMS
  Small systems (MS-DOS, Macintosh, OS/2)

- Development environment—We don't yet have the complete picture, but we believe that commercial programmers require an integrated edit, compile, debug, and test environment, plus sophisticated configuration management. These programmers will accept a more structured CASE environment than will traditional technical UNIX programmers. They are interested in reducing application time-to-market and creating applications that are easy to maintain.
- Individual hardware—Commercial programmers are often using character-cell terminals (CCTs) and personal computers (PCs). A percentage of them are moving toward workstations for the future.

Technical programmers

The secondary audience of technical programmers on OZIX can be characterized as follows:

- Applications—Technical programmers are developing applications such as lab monitoring sytems, process control, and scientific modeling.
- Software experience—They are experienced in application development, and have used some flavor of the UNIX operating system. Their primary programming languages are C and FORTRAN.
- Customer category—Technical programmers on OZIX work for end-user companies, for Digital third-party software vendors, and also include Digital engineers developing layered components and products.
- New versus old code—A smaller percentage of programming involves writing new code, either entire new applications or new portions of existing applications. The larger percentage of the programming involves migrating existing code from other platforms.
- Standards—OZIX programmers are very concerned with adherence to standards. A small percentage of the time they may take advantage of added-value features, but they isolate nonstandard code into separate modules.
- Interoperability—OZIX technical applications need to interoperate with the following environments, from most prevalent to least:

  ULTRIX/OSF
  UNIX systems from other vendors (mostly BSD)
  VAX/VMS
  Small systems (MS-DOS, Macintosh, OS/2)

- Development environment—Technical programmers require an integrated edit, compile, debug, and test environment, plus sophisticated configuration management. They are interested in reducing application time-to-market and creating applications that are easy to maintain. These programmers prefer a CASE environment that is not too rigidly structured.
- Individual hardware—Technical programmers are using a variety of hardware, including character-cell terminals (CCTs), personal computers (PCs), and bitmapped screens (either workstations or windowing terminals).

11.3 Added Value

The OZIX application programming information set adds value in the following ways:

- The programming information set provides easy and direct online access to the information.

  All of the information is available online. The online documentation system provides immediate access to necessary information, often in a context-sensitive fashion from the programming environment.

- The information set is integrated with the programming environment.

  Documentation is directly tied to any aspects of the programming environment that have a windowing interface.

- OZIX documentation provides a comprehensive set of task-oriented information.

  A wealth of task-oriented information is provided, giving the programmer information that is directed at the specific tasks they are performing.

- The information set documents added-value features of the software.

The information set describes such added-value features as enhanced application performance tools, multithreading, and industry-leading compilers.

11.4 Strategy

The strategy for the OZIX application programming information set encompasses several areas: online presentation strategy, hardcopy strategy, and the relationship to other documentation. The following subsections describe these areas.

11.4.1 Online Presentation Strategy

Our long-term strategy is aimed at programmers using bitmapped screens (workstations or windowing terminals), but in the short term, it is also important to address character-cell terminals.

The online documentation is organized according to programming tasks. Within each topic, the information is divided into several subtopics or modules.

Bitmapped screens

All of the OZIX application programming documentation is available online from bitmapped screens. There are multiple ways to enter the online documentation system, but once a programmer accesses any portion of the programming documentation online, they can use the online documentation system to navigate to any other portion of the documentation.

The online programming information is accessed in the following ways:

- If a component of the programming environment has a windowing interface, the documentation is tightly integrated with the application and is accessed in a context-sensitive fashion.
- Language and routine syntax reference information is accessed in a context-sensitive fashion from the text editor(s) that support this feature.
- All the programming information is accessible through the main interface to the documentation system. From the main interface the programmer can perform a search, browse a topics list, and so on.

  The "main interface" refers to the non-context-sensitive way to enter the online documentation system. The actual interface is TBD. The user might select an entry such as "documentation" from an applications menu, or click on a documentation or help icon, or issue a command.

Character-cell terminals

Programmers using character-cell terminals will not be able to access the complete programming information set in an online fashion. Reference information will be available in traditional reference page (man page) format. They will need to use hardcopy documentation or a bitmapped screen to access the conceptual and task-oriented material online.

11.4.2 Hardcopy Strategy

The hardcopy strategy for the OZIX application programming documentation is the same as that for the overall OZIX information set. As with the rest of the information set, hardcopy documentation will be produced from the same source files used for online information.

Like the online information, the hardcopy information is organized according to programming tasks. It is likely that the information for each of the tasks will represent a separate volume or book.

11.4.3 Relationship to Other Documentation

The OZIX application programming information set has the following relationships to other documentation:

- Many of the OZIX programming environment features are in common with the ULTRIX environment. There is an attempt to share documentation with ULTRIX wherever possible.
- Programmers use system features that are common to many types of users, and not just specifically provided for programmers. Programmers need access, therefore, to general user information (for example, overall security concepts, and file manipulation commands such as *ls*).
- Programmers also need access to information in the system and network administration information set, such as networking concepts. The online documentation software permits easy and direct access to this complementary information, regardless of the fact that it resides in another portion of the OZIX information set.
- To support customers, this information set is used by field support personnel, and therefore, complements the support information set.
- Many of the references pages (man pages) document features that are specified by the OSF. It is likely that we will use the OSF reference pages as a basis, and modify them where appropriate to meet the needs of OZIX.
- Information on writing AIA-compliant applications is common with VAX/VMS and ULTRIX. This information overlaps with those documentation sets.

11.5 Application Programming Information Set Summary

The OZIX application programming information set provides conceptual, procedural, and reference information to programmers writing applications on OZIX.

The programming information set is a combination of common and unique documentation. To reinforce the message of a programming environment that adheres to standards and is consistent across Digital open systems platforms, every effort is made to provide common documentation with ULTRIX wherever possible. Where OZIX provides added value beyond other open systems platforms, we provide unique documentation.

Approximately 90% of the programming documentation is common and is written at OSP/DECwest, OSP/Spitbook, or OSP/Palo Alto. Approximately 10% of the programming documentation is unique to OZIX and is written at OSP/DECwest. Note that these percentages are *strongly* dependent on engineering adhering to common functionality, and on the implementation of a successful common documentation strategy.

The programming information set is organized into broad topics which relate to the tasks performed by programmers. These topics form the overall structure of the online documentation modules for programming. Table 6 lists the topics in the OZIX application programming information set, together with their estimated page counts.

Each topic can include conceptual, procedural, and reference information. Many of the traditional routine reference pages (man pages), therefore, are included in a particular topic in the table, rather than being a line item of their own. (The exceptions are the miscellaneous routines that are not part of another topic at this point.) The static structure of the table, though, does not reflect the flexibility available to the programmer with the online documentation system. When reading information online, the programmer can access reference material either through cross-references in procedural or conceptual modules, or in a more component-oriented fashion with the index.

Table 6: OZIX Application Programming Information Set

Topic	Page Count[1]	OSP/DECwest Writers	Other Writers
Introduction to the OZIX Application Programming Environment	50	tbd	-
Building Applications	345	Bill Muse	OSP/Spitbrook
Writing Applications for OZIX			
Writing Portable Applications	50	Bill Muse	-
Writing Application Interfaces	30 + tbd[2]	tbd	OSP/Palo Alto
Writing Distributed Applications	940	tbd (put online)	OSP/Spitbrook
Writing Multithread Applications	250	Liz Hunt	CMA writer
Writing Secure Applications	100	tbd	-
Writing International Applications	70	tbd (put online)	OSP/Spitbrook
Writing Graphics Applications	500	tbd (put online)	OSP/Spitbrook
Writing CDA Applications	800	tbd (put online)	OSP/Spitbrook
Using INGRES, SQL	300	tbd (put online)	Third party
Porting Applications to OZIX	200	Bill Muse	-
Improving Application Performance	60	Liz Hunt	OSP/Spitbrook
Other reference material	500	tbd	OSP/Spitbrook
Using C	630	Steve Johnson	-

[1]The page-count definition may require redefinition to present a more accurate appraisal of the online documentation development. These page-count estimates are based upon traditional, hardcopy algorithms.

[2]The page count of DECwindows, OSF/Motif, and X documentation is TBD.

The following list provides a brief description of the documentation for each topic. For more detailed information, see the individual documentation module plans.

- *Introduction to the OZIX Application Programming Environment*

  This module describes the system from a programmer's point of view. It provides an overview of the utilities, tools, and programming interfaces that are available. It highlights the features and benefits of the system that make it particularly suited to application design and development.

  This module includes summaries and overviews, rather than straight procedural or reference information.

- *Building Applications*

  This topic is really a collection of modules that describe the mechanics of putting together an application. (Note that *writing* application code is covered in another module.) These modules include the following topics:

  — Managing source code

  — Debugging applications

  — Using the message facility

  — Preparing software for distribution

  — Building large applications

This set of modules includes procedural information on the tasks, reference information on the tools and utilities, and some conceptual information.

Of the 345 pages, 225 pages are common with ULTRIX and are written at OSP/Spitbrook, and 120 pages are new pages written at OSP/DECwest. A minimum amount of effort is required to modify the 225 common pages to be used with the online documentation system.

- *Writing Applications for OZIX*

  This is a large topic, so it is divided into the following modules:

  — *Writing Portable Applications*

  This module describes how to write applications that are portable across multiple platforms. This module addresses various levels of portability, such as X/Open compliance, OSF compliance, and AIA compliance.

  This module is related to the module on porting applications to OZIX, but this module addresses writing portable applications from scratch, as opposed to porting existing applications.

  This module includes guidelines on portable application design and reference information indicating which features are supported on which platforms.

  — *Writing Application Interfaces*

  This module describes how to create user interfaces for applications on OZIX. DECwindows, OSF/Motif, X11, and *curses* are included. The text for this documentation module is shared with ULTRIX. The majority of the documentation is written at OSP/Spitbrook and OSP/Palo Alto, and the page count is TBD. A small section on adding online documenation to an application is written at OSP/DECwest (approximately 30 pages).

  This module includes task-oriented material on creating user interfaces, reference information on the user interface tools and routines, and style guide information.

  — *Writing Distributed Applications*

  This module describes how to write applications for the distributed environment, including how to use DECrpc, the X/Open Transport Interface (XTI), the Berkeley Socket Interface, Packet filters, and DECnet-OZIX.

  The current thinking is that the text for this documentation module can be shared with ULTRIX because the product functionality is equivalent. The estimate of 940 pages is the current page count for these topics in ULTRIX. The documentation effort required on this module for OZIX is to modify the information so it can be displayed online.

  This module includes procedural information on distributed applications and reference information on the routine interfaces.

  — *Writing Multithread Applications*

  This module describes how to write applications on OZIX that employ multiple threads of execution within a single address space. Multithread applications are written using the portable services provided by the Concert Multithread Architecture (CMA). Examples of topics in this module include creating and terminating threads, scheduling threads, using handles, and thread synchronization.

  This module includes conceptual information on such topics as objects and handles, procedural information on using multiple threads of execution, and reference information on the CMA routines.

The reference material is shared with the ULTRIX documentation set, and potentially any other platform that implements CMA (approximately 150 pages). The OZIX documentation group will need to apply a minimum amount of effort to these pages to enable them to be displayed online. The conceptual and procedural information should also be shared, but may be written at OSP/DECwest if it is not first available on another platform (approximately 100 pages).

— *Writing Secure Applications*

This module describes how to design applications that must execute in a secure environment. This module outlines the security considerations that must be addressed, and describes how to use the application interface to the security subsystem to perform auditing, quota management, and convert channel control. This module also describes how to install new access control models, new access control classes, and new security authenticators.

This module has cross-references to the general user and system administration information sets for information on general security concepts and user-level security functions.

This module includes secure application design considerations, procedural information on using the security subsystem interfaces, and reference information on the security routines.

This documentation module is unique to OZIX and is written at OSP/DECwest.

— *Writing International Applications*

This module describes how to write or adapt applications to meet international requirements, such as those of multiple local languages and the specific character sets associated with them. This module describes the tools and functions that enable the internationalization of programs and the environment in which they operate.

This module consists primarily of conceptual and procedural information, with pointers to the associated reference pages.

The text for this documentation can be shared with ULTRIX. The documentation effort required on this module for OZIX is to modify the information so it can be displayed online.

— *Writing Graphics Applications*

This module describes how to use GKS and PHIGS. This module is common to ULTRIX and OZIX, and is written at OSP/Spitbrook. A minimum amount of effort is required to modify the common documentation for online use.

This module includes procedural and reference information.

— *Writing Compound Document Architecture Applications*

This module describes how to use the DDIF support and DDIF viewer routines. This module is common to ULTRIX and OZIX, and is written at OSP/Spitbrook. A minimum amount of effort is required to modify the common documentation for online use.

This module includes procedural and reference information.

- *Using INGRES, SQL*

This module describes how to use INGRES and SQL support in OZIX applications. This module is common with ULTRIX, and is written by a third-party vendor.

- *Porting Applications to OZIX*

  This module describes the nuts and bolts of porting an existing application to OZIX. It helps a programmer identify the usually small portions of an existing application that are not portable to OZIX, and instructs them how to adjust the code for OZIX. This module contains language-independent and language-dependent information, and addresses migrating from a variety of platforms.

  This module primarily consists of procedural information on how to move an application to OZIX.

  The estimated page count of 200 pages may be high. For comparison, though, the well-received *PMAX Migration Guide* written by ESG was over 300 pages. This module is unique to OZIX and is written at OSP/DECwest.

- *Improving Application Performance*

  This module describes how to analyze the performance of an application using the OZIX performance tools. It also provides methods for using the performance analysis data to identify and eliminate performance bottlenecks in an application.

  This module includes procedural information on analyzing and improving performance, and reference information on the application performance tools. This module is mostly common with ULTRIX and is written at OSP/DECwest.

- Other reference material

  Some routine reference material is not included in the above topics, and for now, is collected here. These topics include system calls, general purpose routines, math routines, and primitive dbm routines.

  This module consists of reference material, and is common with ULTRIX.

- *Using C*

  This module provides complete reference information for the C language on OZIX, as well as task-oriented information on using the C language and compiler. This module is common with ULTRIX, and is written at OSP/DECwest.

12 OZIX Software Support Information Set

This section describes the information set that is provided for OZIX software support. It includes an overview of the OZIX software support environment, an audience definition, the added value that the support information set specifically provides, the online and hardcopy documentation strategy, and a summary of the software support information set.

The software support information set is intended for use by both customers and support personnel. License management determines which modules of the software support information set are able to be viewed by customers.

12.1 Software Support Environment Overview

The software support environment provides support personnel with information to install, maintain, and troubleshoot the OZIX operating system and related applications. Comprised of information that emphasizes a functional approach, support documentation allows support personnel to install the OZIX operating system easily, or restore a malfunctioning operating system quickly.

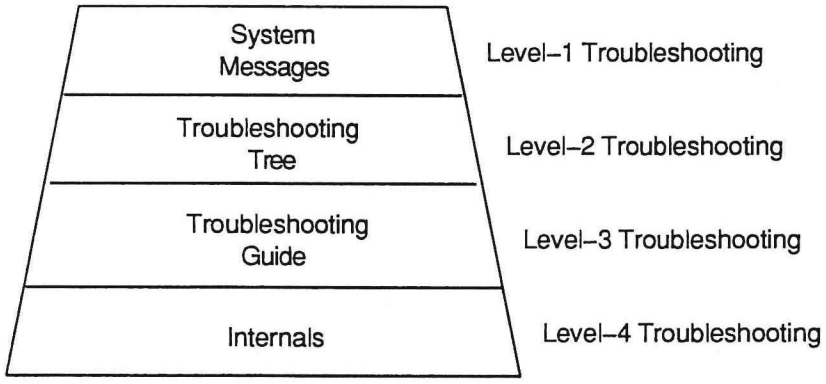
To permit easy installation, a functional approach provides complete steps for the installation of the OZIX operating system and applications. These functional steps range from a delineation of prerequisites to verification that the installation is complete.

To permit prompt restoration, a functional approach provides specific steps that identify the problem and return the software to a normal operating condition. Where a functional approach does not apply, as in problems at the internals level, conceptual information provides the necessary detail for support personnel to understand the normal operation of each component of the operating system. Conceptual information provides the foundation for functional information, and is linked or cross-referenced extensively to specific, known problems.

The structure of the software support environment consists of information for installation and four levels of maintenance. This structure reflects the activities of support personnel at various support levels, such as at the customer site or the Customer Support Center. Additionally, because of a modular format, this structure allows for use of support information by customers, as determined by the License Management Facility (LMF).

Installation addresses a normal sequence of events, leading to a complete, successful installation of the OZIX operating system. Troubleshooting, however, addresses abnormal occurrences within a normal sequence of events. To correct malfunctions quickly, the software support environment approaches simplistic problems at a high level (level 1) and complex problems at a low level (level 4). Figure 2 shows the troubleshooting structure of the software support environment.

Figure 2: OZIX Software Support Environment Troubleshooting Structure



- Level-1 Troubleshooting

  Level-1 troubleshooting includes top-level information to quickly identify and define specific, known problems. *OZIX System Messages* provides this information, with cross-references to level-2 information as needed. OZIX System Messages includes all error messages generated by the operating system, in addition to symptoms for known problems.

- Level-2 Troubleshooting

  Level-2 troubleshooting includes procedural information to quickly resolve specific, known problems. The *OZIX Troubleshooting Tree* provides this quick-reference information, with cross-references to level-1 and level-3 information as needed. The *OZIX Troubleshooting Tree* includes

procedures for each system message defined within *OZIX System Messages*, in addition to procedures for known problems.

- Level-3 Troubleshooting

  Level-3 troubleshooting includes detailed, conceptual information that directly supports the level-2 procedural information, allowing abstract resolutions to be developed, as needed. The *OZIX Troubleshooting Guide* provides this information, with cross-references to level-2 and level-4 information.

- Level-4 Troubleshooting

  Level-4 troubleshooting includes the internals information needed to resolve problems that cannot be corrected by higher levels of troubleshooting. The internals information is primarily conceptual, providing the foundation for a general understanding of the system. *OZIX Internals* provides this detailed information, with appropriate cross-references to higher levels of documentation. Source code also may be available to complement *OZIX Internals*, depending upon LMF and the needs of support personnel.

12.2 Audience

The following Digital personnel use the OZIX software support environment information set:

- Field Service Engineers, first- and second-level support
- Customer Services support staff
- Customer Support Center staff
- Software Services personnel
- Design engineers
- Support engineers
- Manufacturing personnel

12.3 Added Value

The OZIX software support environment information set adds value in the following ways:

- The support information set provides quick and accurate documentation to support personnel, minimizing computer down time.
- The online documentation system provides immediate access to needed information, eliminating time spent accessing information through hard copy.

  By linking OZIX engineering applications such as OZIX installation or the Online Diagnostic Monitor (ODM) to the documentation, it is possible to access needed information as problems occur during the execution of each application.

12.4 Strategy

The strategy for the OZIX software support information set encompasses several areas: online presentation strategy, hardcopy strategy, the relationship to other documentation, and the relationship to training. The following subsections describe these areas.

12.4.1 Online Presentation Strategy

The OZIX software support environment information set will be developed for an online presentation, linked to two OZIX engineering applications: OZIX installation and ODM. This online presentation includes the use of context-sensitive help and DECwindows-style help to access information directly related to a specific part of the engineering applications.

The content of the documentation is modular in structure, enabling easy, quick referencing.

Engineering development will provide a command-line interface for OZIX installation and ODM, as opposed to a graphical interface, for Version 1. A graphical online documentation presentation with linking to OZIX engineering applications will be implemented in Version 2.

Online access to the OZIX software support environment information set is determined by the License Management Facility (LMF). Through a facility to be provided subsequent to Phase 1 exit, LMF will isolate Digital-internal information from unlicensed customers. OZIX internals information is available only to Digital support personnel.

12.4.2 Hardcopy Strategy

The hardcopy strategy for the OZIX software support environment information set is the same as that for the overall OZIX information set. As with the rest of the information set, hardcopy documentation will be produced from the same source files used for online information.

Although support documentation emphasizes an online approach, quality will not be sacrificed for hardcopy development to accomplish online development. When hard copy is the primary deliverable, a modular approach will still be taken for consistency with the online approach.

12.4.3 Relationship to Other Documentation

The relationships among the OZIX software support environment information set, the OZIX system management environment information set, and the OZIX application programming environment information set are integrated and interdependent. In addressing issues and problems that customers encounter, support personnel may use the entire OZIX software information set; however, correcting OZIX internals issues and problems requires a narrower definition, which is accomplished through support documentation.

Because a functional approach often crosses task definition boundaries, a distinct separation is difficult. One virtue of online documentation is that these interdependencies become transparent to the user, making a functional approach feasible.

12.4.4 Relationship to Training

The relationship between the OZIX software support environment information set development and training addresses a number of areas: online documentation, review strategy, and feedback from training personnel. One basic premise in the development of the OZIX software support environment information set is that it should minimize the development of training documentation.

The relationship between online documentation and training merges with computer-based instruction (CBI). The OZIX software support environment information set provides linking capabilities for a CBI application.

The relationship to training includes a review of the OZIX software support environment information set during all phases of development, to ensure compatibility with CBI and to ensure an appropriate level of content. This coordination is important to minimize the repetition of information between training courses and the documentation.

The relationship to training includes training personnel providing feedback to writers on both the online interface and the content of the OZIX software support environment information set. Through this interaction with training personnel, writers can gather initial comments for corrections and improvements to usability and content.

12.5 Software Support Information Set Summary

The OZIX software support environment information set includes information specifically for support personnel. Table 7 lists the topics in the OZIX software support environment information set, together with their estimated page counts.

Table 7: OZIX Software Support Environment Information Set

Topic	Page Count[1]	OSP/DECwest Writers
OZIX Installation	50	Bill Talcott
OZIX System Messages	200	Bill Talcott
OZIX Troubleshooting Tree	200	Rob Shuster
OZIX Troubleshooting Guide	350	Rob Shuster
OZIX Internals	600	Gordon Furbush

[1]The page-count definition may require redefinition to present a more accurate appraisal of the online documentation development. These page-count estimates are based upon traditional, hardcopy algorithms.

- *OZIX Installation*

  OZIX Installation includes information and procedures for the installation of the OZIX operating system. Installation includes the following elements:

  — Prerequisites

  — OZIX operating system installation

  — Installation verification

  — Default responses during installation

  — Installation-to-system-management interface

- *OZIX System Messages*

  OZIX System Messages includes information needed to understand each system message, and necessary cross-referencing information to the *OZIX Troubleshooting Tree*. Additionally, *OZIX System Messages* provides other information related to specific, known problems defined within the *OZIX Troubleshooting Tree*. This topic is the first level of four in the troubleshooting series, preceding the *OZIX Troubleshooting Tree*. *OZIX System Messages* includes the following elements:

  — Structure of information modules for system messages

  — System messages

  — File names for system messages

  — Other symptoms

- *OZIX Troubleshooting Tree*

  The *OZIX Troubleshooting Tree* includes text and graphical information for troubleshooting specific malfunctions in the OZIX operating system. Through a troubleshooting tree format, users are able to address a specific problem and follow a predefined series of steps for correction of the problem. This topic will evolve as problems with the operating system are understood; however, the initial set of information is based on problems found during analysis of the ULTRIX-32 Customer Support Center (CSC) database, during usability testing, and during field test. Subsequent information will be a product of information gathered by the Customer Services organizations on the OZIX operating system. Steps to resolve system messages will be included within this topic also.

  The structure of the topic is preliminary, but should include information that introduces the troubleshooting method, describes use of the topic, provides any necessary cross-referencing to *OZIX System Messages*, and provides any supporting information required to fully understand the troubleshooting scenario. This topic is the second level of four in the troubleshooting series, preceding the *OZIX Troubleshooting Guide*.

- *OZIX Troubleshooting Guide*

  The *OZIX Troubleshooting Guide* is a reference guide that provides an in-depth, detailed description of the *OZIX Troubleshooting Tree*. This topic is the third level of four in the troubleshooting series, following the *OZIX Troubleshooting Tree* and *OZIX System Messages*.

  The structure of this topic is preliminary, but should closely follow the structure of the *OZIX Troubleshooting Tree*. This topic should provide cross-reference capability to *OZIX System Messages* as well. When this topic fails to resolve a specific problem, it should permit an easy transition to *OZIX Internals*.

- *OZIX Internals*

  OZIX Internals includes detailed information on OZIX internals from both a functional and a component point of view. This topic provides users with a detailed description of the components of the OZIX operating system and the way in which these components function together. This topic is the fourth level of four in the troubleshooting series, following the *OZIX Troubleshooting Guide*. Elements described by this topic include the following:

  — OZIX operating system overview

  — Base System Architecture

  — Application programming interfaces (APIs)

  — Executive subsystems

  — Nub

  — Networking protocols

13 OZIX Schedule

This section summarizes the project schedule, the documentation milestones, and the production schedule for completing all phases of the OZIX project by FRS. Specific schedules for each documentation module are available in the corresponding documentation module plan.

13.1 Project Schedule

Table 8 shows the project schedule for OZIX Version 1.

Table 8: OZIX Version 1 Project Schedule

Phase	Date
Phase 1 Exit	November 17, 1989
Begin external field test	July 1, 1991
FRS	December 31, 1991

13.2 Documentation Milestones

Table 9 shows the schedule for OZIX documentation, based on the OZIX Version 1 project schedule.

Table 9: OZIX Documentation Milestones

Milestone	Date
Preliminary master documentation plan available	August 1989
Phase 1 Exit	November 1989
Preliminary individual documentation module plans available	December 1989
Begin writing documentation modules	January 1990
Customer site visits and polling	February 1990
Final master documentation plan approved (signoff)	March 1990
Final individual documentation module plans approved (signoff)	March 1990
Usability lab testing	April 1991
Begin field test production	April 1991
Field test draft distribution	No later than June 1991
Begin external field test	July 1991
Begin final production	July 1991
Field test site visits and polling	August 1991
Begin submission to SSB and translation	September 1991
Complete submission of modules to production	September 1991
Complete final production	October 1991
Complete submission to SSB and translation	November 1991
Complete field test	December 1991
Documentation released for distribution	December 1991
FRS	December 1991

The following subsections describe the OZIX documentation work to be completed for each of the major project milestones.

13.2.1 Phase 1 Exit

The *OZIX Master Documentation Plan* is available for the Phase 1 exit review. Individual documentation module plans for each OZIX module are available when completed in the following public directory: decwet::guest1$:[ozix.v1.docplans]. Each documentation module plan includes the writing schedule for the module, production information, and a detailed outline of the module.

13.2.2 External Field Test

Prior to final production, field test copies of each OZIX module are available for distribution to external field test sites. Field test documents are delivered online as well as in hardcopy form. Field test release notes are also produced.

During the period between field test and SSB submission, the OSP/DECwest production group produces the final masters for each hardcopy manual. Because this period is short, and because of the time required for final production, field test comments may not be incorporated into the Version 1 final masters. If this is the case, and it is determined from field test comments that information critical to the understanding of the system is missing or incorrect, this information is included in the appropriate release notes; otherwise, field test comments are incorporated into the next release of the documentation.

13.2.3 Final Production and SSB Submission

All final documents are delivered in both hardcopy and online formats.

The OSP/DECwest production group produces final masters for all OZIX manuals, including all illustrations, in camera ready format for SSB submission. Manual submission is staggered to complete the submission eight weeks prior to FRS, except for release notes, the master glossary, and the master index, which are submitted to the SSB at the time of the software submission.

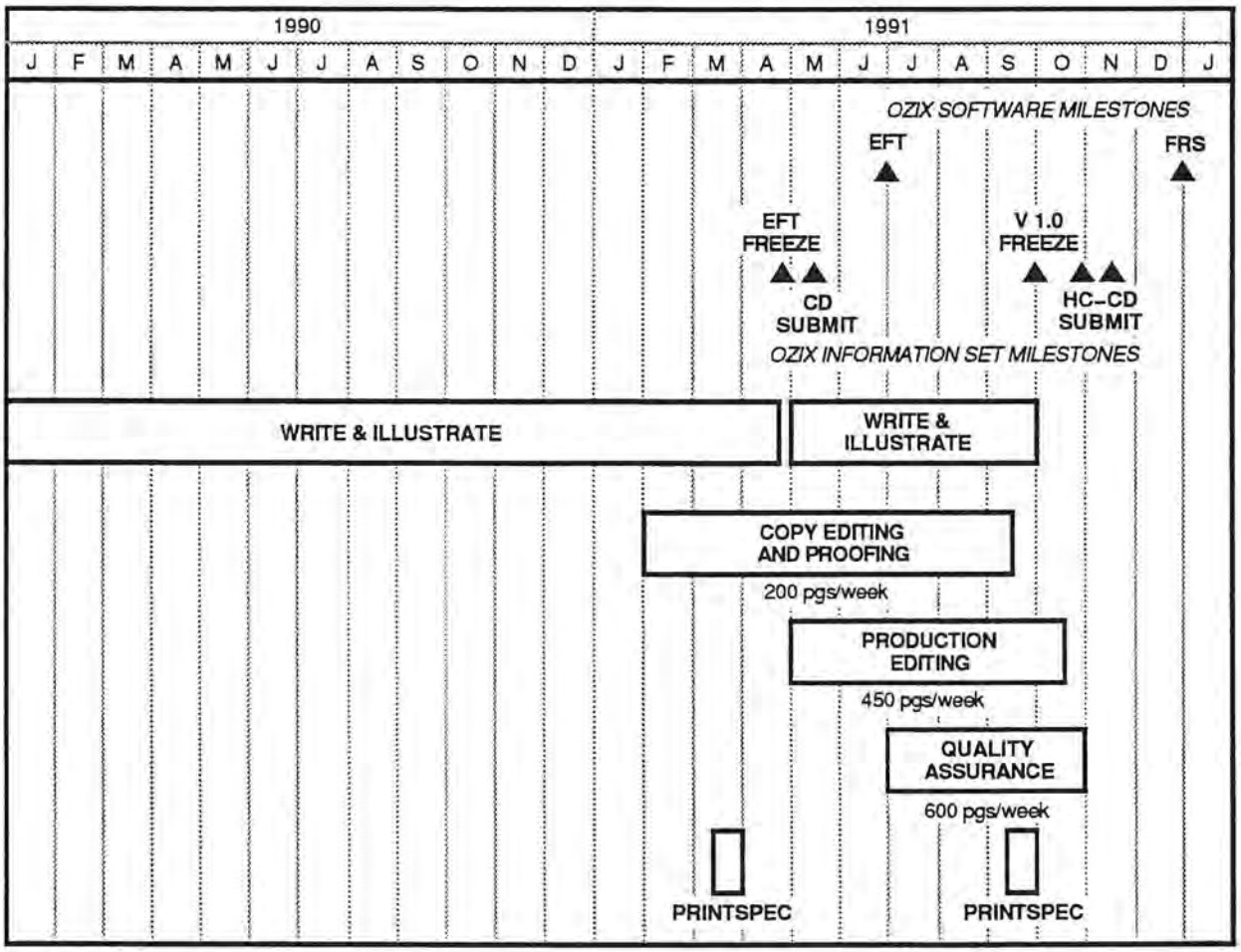
The SSB is responsible for printing, stocking, and distributing the OZIX information set by FRS. In addition, the SSB provides production and distribution of the OZIX information set on CDROM in accordance with DEC STD 073.

13.3 Production Schedule

Figure 3 shows time lines representing the OZIX production schedule. It is important to note the following points:

- EFT freeze and V1.0 freeze dates indicate the last day changes may be entered into a book.
- A pages-per-week count is included with the copy-editing and proofing, production-editing, and quality-assurance phases. It is important that work is submitted to the production group in approximately these volumes.
- The volume of work submitted to the production group multiplied by the number of weeks shown on the schedule often totals more than the total number of pages in the information set. This is due to some tasks requiring repetition.

Figure 3: OZIX Production Schedule



14 OSP/DECwest Writing Resources

This section summarizes the technical writing resources needed by OSP/DECwest to produce the OZIX Version 1 information set, including workload estimates, staffing requirements, and current resources.

14.1 Workload Estimates

The number of pages currently estimated to be written at OSP/DECwest for the general information set is 560. This figure includes only the OZIX-specific user information produced to supplement the common core of user information to be developed by other OSP groups. Note that common user information written elsewhere may require modification for online use, which will increase the OSP/DECwest workload estimate.

The number of pages currently estimated for the system administration information set is 1600. This figure includes all OZIX network administration information, but does not include the descriptions of any implementations of networking protocols (this is OZIX internals information).

The number of pages currently estimated to be written at OSP/DECwest for the application programming information set is 1340. Another 3485 pages of common programming information written elsewhere may require modification for online use. Note that these numbers are heavily dependent on a high percentage of documentation being shared with other ULTRIX family products. If some of the common pages become unique to OZIX, the number of new pages to be written at OSP/DECwest increases considerably.

The number of pages currently estimated for the software support information set is 1400.

Therefore, based on current estimated page counts, the total page count for OZIX documentation to be written at OSP/DECwest is 4900:

```
560 pages + 1600 pages + 1340 pages + 1400 pages = 4900 pages
```

14.2 Staffing Requirements

OSP/DECwest staffing requirements are based on the assumption that one writer can produce 30 finished pages of documentation per month. This reflects the total time commitment over the course of the writing project, including researching, outlining, drafting, reviewing, editing, revising, indexing, and submitting the final module to production.

The introduction of online documentation development redefines this assumption; however, scheduling data for writing online documentation is presently unavailable, limiting the current estimate to a hardcopy estimate. As the online documentation development is better understood, that redefinition will be provided here.

Assuming that all OZIX manuals must be in stock at FRS, scheduled for December 31, 1991, and allowing eight weeks for printing and stocking, the effective date of submission to the SSB is October 31, 1991. Therefore, all manuals must be signed off and in production by September 30, 1991, the Version 1 production freeze date (see Figure 3).

Although writers can begin writing documentation modules in January 1990, they will continue to spend approximately one-third of their time finalizing individual documentation module plans and one-third of their time reviewing and editing engineering specifications through March 1990. Writers can begin full-time writing in April 1990. Therefore, the OSP/DECwest writing team has 19 months to complete the writing effort: one-third of the period from January 1990 through March 1990 (effectively 1 writing month) plus the period from April 1990 through September 1991 (18 writing months).

Using the assumption that one writer can produce 30 finished pages of new documentation per month, and using the average effective writing time per writer of 19 months, one writer can produce, on average, 570 finished pages of documentation over the course of the project.

Therefore, 8.6 writers are needed to plan, write, and produce the OZIX Version 1 documentaion to be written at OSP/DECwest:

```
4900 pages / 570 pages per writer = 8.6 writers
```

In addition, for the 3485 pages of common programming information that may require modification for online use, the assumption is that one writer can modify 100 pages of documentation per month. Using the average effective writing time per writer of 19 months, one writer can modify, on average, 1900 pages of common documentation over the course of the project.

Therefore, 1.8 writers are needed to modify the common programming information for online use:

```
3485 pages / 1900 pages per writer = 1.8 writers
```

The total number of writing resources needed to plan, write, and produce the OZIX Version 1 information set, therefore, is 10.4 writers:

```
8.6 writers + 1.8 writers = 10.4 writers
```

As the online documentation development is better understood, resource estimates may require modification, depending upon the accuracy of current estimates. It is anticipated that online documentation development may require additional resources, but not fewer. Any redefinition of effort will be based upon updated algorithms.

14.3 Current Resources

There are currently 10 people in the OSP/DECwest writing group. Jim Jackson manages the publications group and has full-time management responsibilities. Marcia Aguero, Liz Hunt, and Bill Talcott have half-time project leading and supervisory responsibilities. Cheryl Snyder has full-time usability testing/engineering responsibilities. Therefore, the number of available resources is effectively 6.5 writers.

Because 10.4 writers are needed to plan, write, and produce the OZIX Version 1 information set, OSP/DECwest needs to hire four additional writers to cover all current commitments.

15 OSP/DECwest Production Resources

This section summarizes the graphics, editing, and publishing system support resources needed by OSP/DECwest to produce the OZIX Version 1 information set, including workload estimates, staffing requirements, and current resources.

15.1 Workload Estimates

For production purposes the OZIX information set is divided into two sets:

1. The first set consists of new information created by OSP/DECwest writers. The page count for this set is estimated at 4900.
2. The second set consists of common programming documentation written by other OSP groups. Because the primary focus for delivery of the OZIX information set is online, and many of the books in this set may not be produced for online delivery, production resources must be dedicated to work on these books. The page count for this set is estimated at 3485.

Many production tasks need to be repeated as books change (such as text editing, proofing, and production editing). Therefore, to allow for a second pass on approximately 30% of the information, the production page count for the first set is increased to 6370. Because some books in the second set may already be written for online, the page count for that set remains at 3485.

15.2 Staffing Requirements

Table 10 shows the different tasks performed by the production group and the hours required to complete each task for an average 100-page book containing 15 illustrations. Note that for the second set of online books, only the scheduling, quality assurance, Printspec coordination, production editing, and CD tape backup tasks are performed.

Table 10: Production Task Time Factors

Task	Hours per Book (set 1)	Hours per Book (set 2)	Total for Docset
Scheduling and coordination	16	5	1194
I18N and translation coordination	7	—	448
Copy editing and proofing	30	—	1920
Quality assurance	12	6	972
Printspec and SSB coordination	7	3	550
Production editing	24	12	1944
Illustration	30	—	1920
Graphics	5	—	320
Final master printing	1	—	64
CD tape backup	1	1	98
Total			9430

The production phase of the OZIX information set, including copy editing and proofing, production editing, quality assurance, and Printspec support, covers a 9-month period (see Figure 3). This equates to approximately 1440 hours per person:

```
9 months * 4 weeks per month * 40 hours per week = 1440 hours
```

Therefore, 6.5 people (illustrators and editors) are needed to complete the production work for the OZIX Version 1 information set:

```
9430 hours total / 1440 hours per person = 6.5 people
```

Tasks are generally divided equally between editors and illustrators; therefore, three editors and three illustrators are required.

In addition to editors and illustrators, the production group requires support from dedicated publishing system administrators. Because the OSP/DECwest publications group is using two distinctly different document processing systems (VAX DOCUMENT and *ditroff*), two system administrators are needed.

15.3 Current Resources

There are currently two illustrators, one editor, and one system administrator in the OSP/DECwest production group. Craig Kosak supervises the production group, and is available as a backup for many tasks, but normally is dedicated to full-time administrative responsibilities (Craig also supervises OSPTG).

Because three illustrators, three editors, and two system administrators are needed to complete the production work for the OZIX Version 1 information set, and to provide dedicated publishing system support, OSP/DECwest needs to hire one additional illustrator, two additional editors, and one additional system administrator to cover all current commitments.

16 Risks and Dependencies

This section summarizes the risks and dependencies associated with producing the OZIX information set.

16.1 Risks

The development of the OZIX information set includes the following risks, which require monitoring throughout the development process for necessary alternative action:

- A single database provision for both hardcopy and online documentation may not be feasible, requiring two sources for documentation.
- Development of the online documentation system defined by the OZIX writing group may not be feasible in the OZIX Version 1 timeframe, limiting the functionality of the documentation relative to the operating system, resulting in limited usability of the OZIX information set.
- If engineering does not adhere to current plans for common functionality with other ULTRIX family products, OSP/DECwest resource estimates may increase considerably.
- Resource estimates may require either additional resources or a reduction of effort, depending upon the accuracy of present estimates.

16.2 Dependencies

The development of the OZIX information set includes the following dependencies:

- Writing development is dependent upon the availability and reliability of hardware, software, and documentation tools that provide a productive, dependable, and sophisticated documentation development environment.
- Writing development is dependent upon training in the following areas: modular writing techniques, authoring tools, usability testing, and indexing for online documentation.
- Information set accuracy is dependent upon timely development of project source documentation, the availability of consultation time with developers as required, and writer participation in meetings affecting functionality or product description.
- Information set completeness and the quality and availability of publications tools is dependent upon adequate staffing.
- Information set timeliness is dependent upon adherence to schedules by engineering development, documentation, and production.
- Information set usability is dependent upon the successful sourcing and testing of an accurate and extensive group of users and the availability of an online documentation sytem throughout development.
- Information set quality is dependent upon a successful review strategy.
- Information set translation is dependent upon the decision of the Digital LEGs to translate OZIX information, together with a successful implementation plan, including the timely assignment of a UI engineer by IED and at least one GIA translation group representative in Asia, timely feedback on the documentation plans and drafts, and the timely transfer of information into the translation process.
- Online information retrieval is dependent upon engineering development to provide the necessary hooks to integrate context-sensitive help or DECwindows Help files with graphical interfaces to OZIX applications such as configuration management, diagnostic monitor, and installation.
- Documenting system administration tasks in terms of a windowing interface is dependent upon the availability of the OSF/Motif-based manageable object browser/editor in the OZIX Version 1 timeframe.

- Information set production is dependent upon the use of *ditroff*. Currently, *ditroff* does not offer functionality that allows production of compound and EDMS-compliant PostScript files, nor does it support output in the Bookreader file format. The OSP/DECwest production group must either find a way to build compound document and EDMS functionality into *ditroff* or rely on another group to do the work. In addition, *ditroff* needs to be audited to ensure that it supports all internationalization requirements.
- The OSP/DECwest production group is dependent upon OSPTG to provide Bookreader output functionality for *ditroff*.
- The OSP/DECwest production group is dependent upon SSB to handle printing, stocking, and distribution of the information set on CDROM and in hardcopy format.
- Software support documentation is dependent upon license management to provide a facility that isolates Digital-internal information from unlicensed customers.
- Software support documentation is dependent upon a clear, quick, and efficient method for including changes and providing them to support personnel. This effort must be coordinated with iterative releases of documentation on CDROM.
- The OZIX software support environment must be coordinated with hardware documentation development and with all OSP documentation development to ensure the best possible service support from Digital.