Building Dependable Systems: The OpenVMS Approach

Order Number: AA–PV5YB–TE

March 1994

This handbook describes the principles of building dependable computing systems. It also highlights the dependability features of OpenVMS software, layered products software, and related hardware components.

Revision/Update Information: This manual supersedes *Building Dependable Systems: The OpenVMS VAX Approach*, OpenVMS VAX Version 6.0.

Software Versions: OpenVMS AXP Version 6.1
OpenVMS VAX Version 6.1

Digital Equipment Corporation
Maynard, Massachusetts

March 1994

ZK5709

This document is available on CD–ROM.

This document was prepared using VAX DOCUMENT Version 2.1.

Send Us Your Comments

We welcome your comments on this or any other OpenVMS manual. If you have suggestions for improving a particular section or find any errors, please indicate the title, order number, chapter, section, and page number (if available). We also welcome more general comments. Your input is valuable in improving future releases of our documentation.

You can send comments to us in the following ways:

- Internet electronic mail: `OPENVMSDOC@ZKO.MTS.DEC.COM`
- Fax: `603-881-0120 Attn: OpenVMS Documentation, ZKO3-4/U08`
- A completed Reader's Comments form (postage paid, if mailed in the United States), or a letter, via the postal service. Two Reader's Comments forms are located at the back of each printed OpenVMS manual. Please send letters and forms to:

  ```
  Digital Equipment Corporation
  Information Design and Consulting
  OpenVMS Documentation
  110 Spit Brook Road, ZKO3-4/U08
  Nashua, NH  03062-2698
  USA
  ```

You may also use an online questionnaire to give us feedback. Print or edit the online file SYS$HELP:OPENVMSDOC_SURVEY.TXT. Send the completed online file by electronic mail to our Internet address, or send the completed hardcopy survey by fax or through the postal service.

Thank you.

Contents

Tables

Preface

Intended Audience

This manual is intended for systems analysts, application designers, system managers, network managers, and database administrators who are responsible for setting up and maintaining the dependable operation of computing systems that run on the OpenVMS™ operating system.

Document Structure

This handbook, *Building Dependable Systems: The OpenVMS Approach*, consists of 11 chapters, two appendixes, a glossary, and an index.

Chapter 1 defines the characteristics of dependable computing systems and introduces key dependability terms and concepts.

Chapter 2 describes how to analyze the dependability requirements of the users of a computing system. The chapter outlines a general approach that can be used to reconcile those requirements with overall business goals.

Chapter 3 offers a method for examining dependability options in the system's environment, hardware, communications, software, personnel, and operational procedures for determining the most effective way to provide improved dependability characteristics.

Chapter 4 examines how to balance your dependability goals with other business considerations.

Chapter 5 describes how the physical environment of a computing system affects its dependability.

Chapter 6 describes how closely the dependability of a computing system is determined by its hardware configuration. A method for analyzing a hardware configuration is provided and several typical configurations are illustrated and explained.

Chapter 7 describes techniques and products that can be used to build dependable networks.

Chapter 8 describes selected Digital™ software products, as well as suggested coding techniques, that contribute to the high availability and performance of software applications.

Chapter 9 provides common sense rules, tools, and techniques for data center operations and system management. The functions discussed in this chapter describe the impact of, and recommendations for, reducing the number of errors and failures caused as a result of system management tasks.

Chapter 10 provides brief descriptions of the services available through the Digital Consulting organization. A summary of the Digital online service connection, DSNlink, is also in this chapter.

Chapter 11 presents a case study of a lights out data center that provides 100% application availability for a Digital Customer Support Center in Colorado.

Appendix A contains evaluation checklists to help you determine whether your computing system meets the minimum requirements for an acceptable computing environment.

Appendix B contains a bibliography of suggested readings for further study.

The Glossary contains definitions of terms used in discussions of system dependability.

Associated Documents

This handbook is not an exhaustive discussion of the product components described within. Therefore, as you use this handbook, you may find it helpful to refer to the documentation sets for the products mentioned. The discussion of most topics in this handbook includes references to specific manuals in component product documentation sets.

Conventions

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

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

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

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

The following conventions are also used in this manual:

boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason (user action that triggers a callback). Boldface text is also used to show user input in Bookreader versions of the manual.
italic text	Italic text emphasizes important information and indicates complete titles of manuals and variables. Variables include information that varies in system messages (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *device-name* contains up to five alphanumeric characters).

UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1 Introduction to Dependable Computing

When your organization cannot conduct its work due to the failure of a computing system component, the consequences can range from merely inconveniencing certain users, to the temporary loss of productivity and revenue, to more severe impacts like harming the organization's long-term financial health or even endangering public safety.

A **dependable computing system** is one that can be counted on to provide services to its users when those services are needed. Beyond the ability to log in to the system, a dependable computing system provides sufficient performance so that users and applications can conduct their work efficiently. Building dependable computing systems is a continuous process that involves complex interdependencies between these components of an enterprise:

- Environment
- Hardware
- Communications
- Software
- Operational procedures
- Personnel

This chapter introduces key dependability terms and concepts and describes different levels of dependability. Subsequent chapters in this handbook provide the methodologies for defining and configuring the proper mix of computing resources and services to meet the business requirements of an organization.

1.1 Levels of System Dependability Requirements

Dependability requirements for most organizations generally fall into four broad (and sometimes overlapping) categories:

- Business functions with computing dependability requirements that can be met by **conventional computing systems.** An example of a conventional computing system is a standalone AXP™ system or a local area network (LAN) of AXP and VAX™ systems.
- Business functions that require uninterrupted computing services, either during essential time periods, or during most hours of the day and most days of the week throughout the year. These dependability requirements need to be met by **highly available computing systems**. An example of a highly available computing system is an environment comprised of VMScluster™ nodes, shadowed disks, uninterruptible power systems (UPS), and database systems that provide sufficient transactions per second (TPS) rates and allow for backup operations when there are active users. Requirements for highly available computing systems typically mean a goal of 24 hours a day, 365 days a year; this measurement is often shown as **24x365**. However, in this

category, rare exceptions can be made for minimal down time to perform certain operations like upgrading the system's hardware or software.

- Business functions that absolutely demand uninterrupted computing services. Functions of this nature are sometimes called **mission critical** because achieving the business's financial goals often depends directly on how well its computing systems support the firm's computerized business applications. These stringent dependability requirements need to be met by **fault tolerant computing systems**, like the Digital™ fault tolerant VAX (VAXft).
- Business functions that absolutely demand uninterrupted computing services and need to be immune from disasters like earthquakes, fires, floods, hurricanes, power failures, vandalism, or even acts of terrorism. These extremely stringent dependability requirements need to be met by **disaster tolerant computing systems**. An example of a disaster tolerant computing system is a single VMScluster that exists as two geographically distant **lobes**. Each lobe of the single VMScluster system is connected by high-speed Fiber Distributed Data Interface (FDDI) fiber-optic cables, with a distance of up to 40 kilometers between the two lobes. Hardware redundancy configurations for each component of the geographically distributed VMScluster system are in place to provide for most disasters.

  Note that a system with requirements for disaster tolerance may not necessarily also need 24x365 operation. In addition, a disaster tolerant environment may not necessarily require a 40 kilometer geographic distance. In situations where disaster tolerance is being considered for computing systems, you should define a **safe distance** based on the environmental factors of the region and a recognition of your business requirements for redundancy.

The business examples in the following list may help you visualize the different levels of dependability requirements:

- Conventional computing

  A firm of tax accountants maintains client tax data on line for (up to) the previous three years. The safety and security of their client on-disk data is important, and a process is in place to perform disk-to-tape weekly backups and nightly incremental backups. Backup tapes older than one week are stored off site. The accountants and their staff typically use their computers during regular business hours (9 a.m. to 5 p.m.). While the director of computing services wants the computing systems to never fail, system outages at night not affecting the integrity of data on the disks are not a significant concern.
- Highly available computing during specific time periods

  A densely populated city uses an extensive network of computers to coordinate the flow of traffic through a grid of intersecting streets in its downtown business district. During the early morning and late afternoon commuting hours, thousands of motorists are aided by the city's computers that control the timing of the overhead traffic lights. City planners know that the traffic flow application is critical during the rush hours. Should the computing systems fail then, the consequences could be significant. For instance, motorists would be inconvenienced, traffic congestion would quickly and adversely affect air quality, and ambulances might have trouble responding to emergencies. If the computers fail at 3 a.m. Sunday and the outage lasts for 10 minutes, the results are still undesirable but are not catastrophic.

The design of computing services for this city's traffic flow application is modeled after the need for uninterrupted services during easily defined time periods. During the rush hours, the dependability requirements are mission critical. Although a level of flexibility is provided during nonpeak hours for hardware or software upgrades and other system management chores, the hardware and software configuration must take into account the peak hour requirements.

- Highly available computing, 24 hours a day, 365 days a year, with minimal exceptions

  A large, international hotel corporation operates a reservation system for its 1,000 hotels worldwide. Reservations can be entered or checked 24 hours a day, 365 days a year. The centralized database of reservations must be backed up while multiple agents are reading existing entries or entering new reservations. Twice a year, system management personnel schedule a limited stretch of time for the system to be brought down for hardware and software upgrades.

- Fault tolerant computing

  At 4:23 a.m. a hospital nurse is watching monitors for patients in the intensive care unit (ICU). The patients and their families rely on the dependability of the monitors, and the monitors are dependent on the computers that run them. For this scenario, a fault tolerant processor, shadowed disks, and database management software that provides run-time journaling, after-image journaling recovery, and online backup features might be used.

- Disaster tolerant computing

  A major banking corporation's computer-based applications can never be allowed to fail. Assume that the cost for *each minute* of down time is ¥50,000. After exploring a number of options, the banking corporation elects to expand and partition its 8-node VMScluster system. Some portion of the VMScluster system will reside 40 kilometers apart, connected by FDDI fiber-optic cables. The two portions of the single VMScluster system are mirror images of the applications and databases that comprise all the corporation's applications. Should a disaster like a building fire affect one site, the other site can continue processing. Shadowed disks provide duplicates of the system and user data, and the VMScluster software helps to alleviate the difficulty of system management tasks for a large and geographically distributed system.

1.2 How Do You Build Dependable Systems?

Computing service interruptions, or down time, annoy everyone involved. Users with even conventional dependability requirements are affected by any unexpected denial of service. System managers, database administrators, network managers, and others responsible for the continuous and predictable operation of computing services dread unexpected down time, of course, because it typically results in an intense scramble to identify the cause of the outage, to restore (where possible) lost data, and to fix the root of the problem so that it will not occur again. Everyone, especially senior management, deplores computer outages because they generate mistrust over the dependability of the applications. Employees start to change their work habits, perhaps not productively, as they wonder, " . . . when will this system go down next?" For organizations that offer their computing services as a product to clients, outages can mean that clients will take their business elsewhere.

1.2 How Do You Build Dependable Systems?

Everyone agrees, then, that unscheduled computing service outages are to be avoided whenever possible. Unfortunately, doing so is not simply a matter of installing redundant hardware. A number of proven, guiding principles exist that have helped computer professionals successfully build dependable systems. The key principles are described in Chapter 2. The following list briefly introduces the principles:

- Plan ahead to prevent problems from occurring, to handle problems when they do occur, and to mobilize additional resources if problems too big to handle immediately occur.
- Go through a formal process of identifying the exact nature of your organization's dependability requirements.
- Go through a formal process of identifying the exact nature of your applications' dependability requirements. An appropriate configuration for application A might be entirely wrong for application B and application C. You may have to segregate the resources needed to support two sets of applications that have differing needs.
- Define, test, implement, and modify as needed to reflect volatile business requirements your strategies for fault prevention, error correction, and failure recovery.
- Provide continuous, top-notch education and training of your computer operations staff.
- *Document* the procedures and policies in your computing operation carefully and extensively. Document the practices that failed, too, so that future employees will not repeat mistakes. Valued employees who leave your organization potentially take with them knowledge and wisdom about the methods used to make things work.
- Establish a process where employees learn each other's jobs, when possible and appropriate. The advantage is that your organization duplicates knowledge and employees can catch each other's mistakes and avoid some problems early on.
- Create a continuous cycle of testing procedures. Perform the tests with diligence.
- Design emergency drills for the staff and run the drills on a scheduled basis, without warnings, and often.
- Use selected vendor hardware products, software products, and services that match your dependability requirements. Assume nothing about the appropriateness of a vendor's product for your business applications. Instead, verify everything.
- Use care and sensitivity to assist those employees who have new responsibilities or now-obsolete assignments, if improving the dependability of your computing systems results in personnel changes. The overall morale of the new "dependability team" may hinge on the fairness of management's decisions during the transition. For example, a staff of 35 lab operators will be cut down to just 5 operators as a data center evolves to a paperless environment that also performs all disk-to-disk backups. Whenever feasible, work aggressively to help the 30 displaced employees find more productive, rewarding jobs in your organization or another area of the company.

- Define the impact of security breaches on your dependability requirements. If you have redundant CPUs and disks, but an intruder can physically or electronically get into your unprotected computer lab, you have not built a dependable system.
- Similarly, protect mission-critical applications against environmental influences such as fires, power failures, and air conditioning equipment. Define your requirements with the following thought in mind, “The worst possible scenario will occur—often.” Prepare contingency plans, too.
- Identify your organization’s constraints (physical site, capital funding budget for equipment and software, organizational, legal, others). Use your knowledge about the constraints to assess the degree to which you can guarantee dependable computing services.

The remaining chapters in this handbook describe the key dependability principles in much more detail. Also see the data center evaluation checklists in Appendix A or in SYS$EXAMPLES (VMS_DEPENDABILITY_CHECKLIST.PS for PostScript® output and VMS_DEPENDABILITY_CHECKLIST.TXT for text output[1]). The checklists may help you determine whether your current computing resources meet the minimum rquirements, as suggested by Digital, for a dependable computing environment.

1.3 Dependability Terms

Unfortunately for readers who are interested in topics about providing continuous computing services, the definition of terms like *availability*, *reliability*, and *fault tolerance* used in computer science textbooks, trade magazines, and vendor product documentation often vary from source to source. This handbook uses these terms as consistently as possible with the majority of sources available.

In the following list, dependability terms are explained:

- A **computing component** is a part of the total computing system around which an arbitrary boundary has been defined. The boundary can be defined at any level. Examples include:
  - An integrated circuit chip
  - An optical disk drive
  - A FORTRAN subroutine
  - A relational database management system
  - The Cleveland branch’s VMScluster system
  - An operator
  - The system manager
  - The electric power utility
- A **fault** is a defect in some component of a computing system. Under certain conditions, such a state would produce an error if the computing component were required to perform its function. Examples include:
  - A bad spot on a magnetic tape
  - A typographical mistake in the program source

[1] These files are available in SYS$EXAMPLES on VAX system disks only; however, the information applies to AXP or VAX environments.

- An unnoticed console message
- A truck hitting a key utility pole

- An **error** is an event during the operation of a computing component that produces incorrect results due to one or more faults. Errors are observed as incorrect responses within a specific computing component. Examples include:
  - A nonmatching checksum while reading data
  - Executing a faulty code path in a program
  - Mounting the wrong magnetic tape
  - The lights flickering
- A **failure** is the inability of a computing component to perform its function correctly due to one or more internal faults whose effects cannot be contained. Failures are observed by the consumers of the computing component's services as incorrect behavior. Examples include:
  - Not being able to successfully read the stored data
  - A program aborting when tested
  - Purging away (deleting) the working version of a program
  - An extended power outage
- **Availability** is the percentage or amount of scheduled time that a computing system provides application service. System availability is only one component of a dependable computing system. For example, the ability to log in to an AXP node or a VAX node means that the OpenVMS™ system is available but not necessarily meeting the needs of its users; application availability meets that need. The following equation is one way to measure availability:

  ```
  Availability (%) = (Total usable hours / Total elapsed hours) * 100
  ```
- **Reliability** is the ability of a computing system to operate without failing. Reliability commonly means the absence of errors or faults in the system's components such that the system is able to serve its users and applications. Reliability can be measured by mean-time-between-failures (MTBF):

  ```
  MTBF (hours) = (Total elapsed hours / (Failure count + 1) )
  ```
- **Fault tolerance** is the ability of a computing system to withstand faults and errors while continuing to provide the required services. A high level of failover transparency (transparent to the system's users and applications) is usually implied by this term. An example of fault tolerance is a shadow set of two disks managed by Volume Shadowing for OpenVMS. If one disk fails, I/O processing continues to the remaining disk member of the shadow set.
- A **dependable computing system** is a system that can be counted on to provide reliable services to its users when those services are needed and do so with sufficient performance. The computing components are created and combined in the manner necessary to provide the required level of trustworthiness. In designing dependable computing systems, the question to ask yourself is, "Is the system dependable enough to meet the needs of my business?" Examples of dependable computing system *components* include:
  - VMScluster systems

  - Fault tolerant VAXft systems
  - Transaction processing (TP) monitors
  - Database management systems
  - Educational and enterprise integration services
  - Site design and facilities management services
  - Uninterruptible power systems (UPS)
- **Fault prevention** is the process of designing and constructing computing components to be free from faults. The goal is zero defects. Examples include:
  - Using CAD/CAM tools for design and manufacture
  - Using structured or object-oriented methods and CASE tools for application creation
  - Providing comprehensive training for operational staff
  - Investing in uninterruptible power systems (UPS)
- **Error correction** is the action necessary to isolate the effects of faults to a specific computing component. The goal is to contain the impact of the problem. Examples include:
  - Using redundant bits in memory arrays for error correction
  - Implementing exception routines to handle unexpected software conditions
  - Producing comprehensive contingency plans for operators
  - Investing in excess air conditioning capacity (via additional, independent units)
- **Failure recovery** is the action necessary to restore the failed computing component to a correctly functioning condition. The goal is prompt return to zero defects. Examples include:
  - Installing spare parts from the stock room
  - Installing the corrected application
  - Signaling the system manager's beeper
  - Cuing the hot standby site to take over
- **Fault management** is the discipline used to engineer systems with a cost-effective balance of fault prevention qualities, error correction capabilities, and failure recovery facilities. Fault management is realized in the implementation of a dependable computing system. It is also a philosophy that is followed during the implementation. Examples include:
  - Designing the VAXft systems with redundant zones to create complete transparency of fault tolerance. Doing so ensures that any major subsystem of the hardware configuration can fail and the running application is not perturbed in any way.

- Implementing robust applications by utilizing a transaction processing (TP) monitor and a database management system. Doing so ensures that application maintenance and database redesign can be performed without interrupting the production environment (by using certain Digital products, described in Chapter 8).
- Cross training operators and rotating assignments periodically. Doing so ensures that sickness and vacation do not leave important operational functions uncovered. It also may result in broader professional perspectives for each employee, in turn leading to better ideas and innovations regarding the operation of computing resources.
- Investing in separate main power supply feeds from more than one power grid. Doing so ensures that local power outages do not impact continued operation.

Please refer to Section 1.4 for conceptual information about dependability. In particular, see Section 1.4.1 for conceptual information about the three primary dependability strategies: fault prevention, error correction, and failure recovery.

1.4 Basic Concepts of Dependability

If you examined all the pieces of a computing system in the finest detail, you would find that each individual component is in one of *three* conditions. The first two conditions are obvious:

- Broken

  A nonfunctioning or incorrectly functioning condition; for example, an RA90 disk device that is not functioning after a disk failure.
- Not broken

  A correctly functioning condition; for example, a software application that is operating without errors.

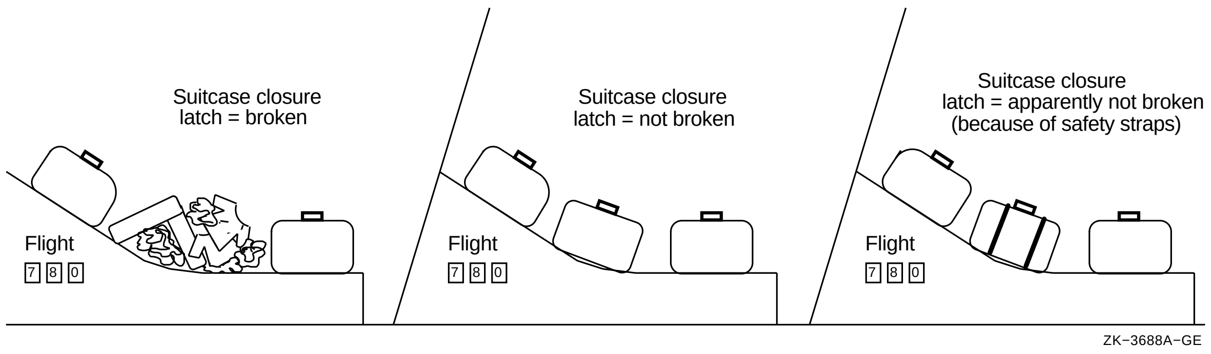
While it may appear that these two conditions are mutually exclusive, dependable systems also make *extensive use* of a third condition. Upon closer examination, this condition is actually the phenomenon of a computing component existing in *both* conditions at once:

- Apparently not broken

  A condition where one or more internal problems exist, but verifiably correct outputs are being produced in spite of the problems. For example, a fault tolerant VAX (VAXft) processor has two separate zones that serve as backups to each other. The “apparently not broken” condition exists when, for instance, zone A of the VAXft has stopped functioning due to a hardware problem, but the redundant zone B is performing normally.

Figure 1–1 illustrates a metaphor drawn from these conditions.

Figure 1–1 Operational Conditions Metaphor


Suitcase closure latch = broken
Flight
7 8 0
Suitcase closure latch = not broken
Flight
7 8 0
Suitcase closure latch = apparently not broken (because of safety straps)
Flight
7 8 0
ZK−3688A−GE

In Figure 1–1, three conditions of the suitcase closure latch are depicted. The left panel clearly shows a latch that is broken; with this metaphor, think of the scattered contents of the suitcase as a *failure*. The middle panel shows a latch that is not broken; it is working properly. The right panel represents an “apparently not broken” condition; the latch has opened, but backup safety straps have prevented a failure (scattered suitcase contents) from occurring.

Other examples of dependability terms drawn from the suitcase metaphor:

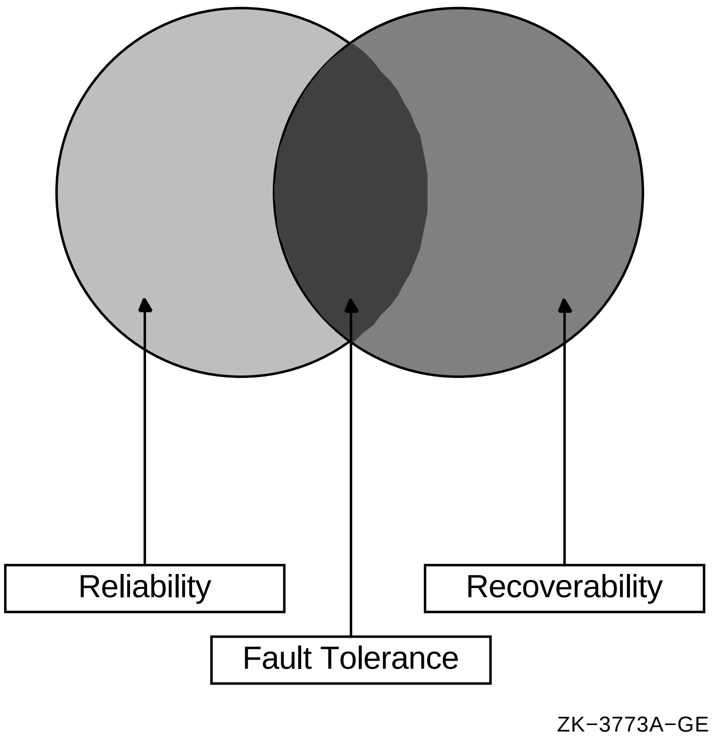
- A **fault** could be a piece of metal on the latch bending out of shape.
- An **error** occurs when the latch opens.
- A **failure** occurs when the suitcase spills its contents.
- **Fault tolerance** is what results because the safety straps contain (correct) the error.

The broken, not broken, and apparently not broken conditions are important dependability concepts because the conditions are factors in:

- Three primary *strategies* for building dependable computing systems
- Three main *characteristics* of dependable computing systems

Figure 1–2 illustrates the main characteristics of dependable computing systems.

Figure 1–2 Characteristics of Dependable Computing Systems



In Figure 1–2:

- The **reliability** characteristic indicates a component's ability to maintain a not broken condition, by virtue of fault prevention strategies.
- The **recoverability** characteristic indicates a component's ability to return from a broken condition to a not broken condition, by virtue of failure recovery strategies.
- The **fault tolerance** characteristic indicates a component's ability to exhibit an apparently not broken condition, by virtue of error correction strategies.

Figure 1–2 reveals that fault tolerance is actually a level of recoverability that is so effective that the computing component (or system) is seen as extremely reliable. That is, it *appears* to never break.

The following sections explain these concepts in more detail. Section 1.4.1 provides another metaphor that may help you understand how these dependability strategies work. Section 1.4.2 expands on the metaphor introduced in Section 1.4.1 and discusses a concept called redundant functional units.

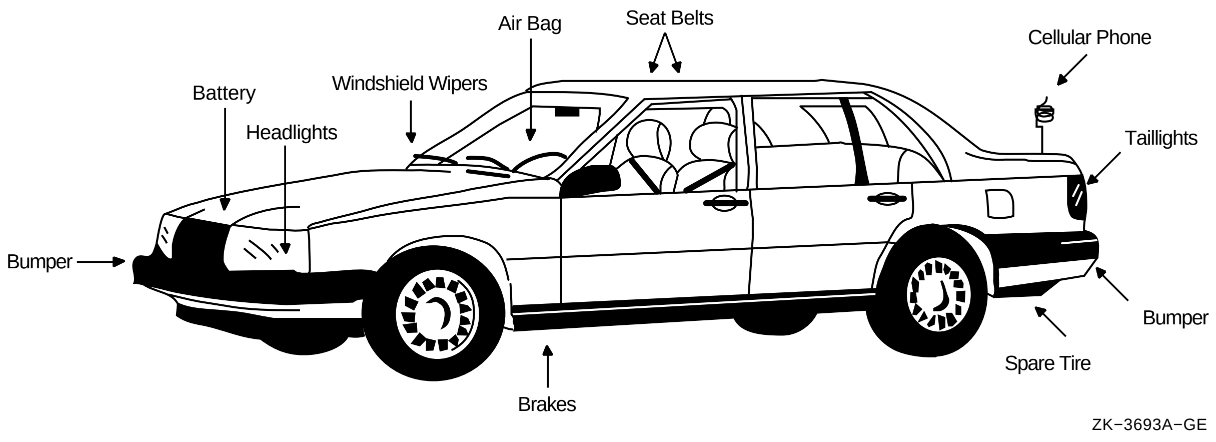
1.4.1 Primary Dependability Strategies

The three essential strategies for building and operating dependable computing systems are:

- Fault prevention: ensure that the component cannot ever break.
- Error correction: when the component is broken, mask that fact.
- Failure recovery: when the component is visibly broken, return it to an unbroken state *quickly*.

To understand these three key dependability strategies, consider the following metaphor of an automobile's components as shown in Figure 1–3. Some of the automobile's standard components, plus specialized safety features and backup systems, combine to create a vehicle that is much more dependable than a vehicle lacking those features.

Figure 1–3 Metaphor of Selected Automotive Components and Dependability Strategies



The identified automobile components correspond to the key dependability strategies in three categories: fault prevention, error correction, and failure recovery.

Under fault prevention:

- A voice-operated cellular phone: by being hands-off, there is less likelihood of the operator losing control of the vehicle.
- A low-fuel indicator on the dashboard: much cheaper than auxiliary fuel tanks, but this feature needs the right environment to work (plentiful service stations).
- Warning signals that are activated automatically as a result of some previously defined condition: examples include a recognizable beeping signal when the operator leaves the vehicle's headlights illuminated, ignition keys in the ignition when the vehicle's motor is off, or the failure to secure seat belts.
- A well-maintained anti-lock braking system: anti-lock brakes help the operator from losing control of the vehicle, especially during emergency deceleration situations. An important factor in the success of this technique for fault prevention is a proper schedule of inspections and repairs to the brake system components.
- Windshield wipers: by keeping the operator's view of the road clear during rain storms, there is greater likelihood of the operator maintaining safe control of the vehicle.
- A car radio tuned to a station that broadcasts traffic congestion reports: alerts the operator to possible trouble areas ahead.

- Headlights and taillights: headlights illuminate the road ahead at night and during inclement daytime weather, increasing the likelihood of the operator maintaining safe control of the vehicle. Similarly, taillights help other operators notice the vehicle, and properly functioning (brake-activated) taillights warn operators in vehicles behind this vehicle when this vehicle is preparing to stop.

Under error correction:

- An auxiliary fuel tank with automatic switch-over feature: when primary fuel tank is empty, a sensor automatically corrects the error and normal operation continues. However, the operator must be made aware that an auxiliary device is now active because the backup system now represents a single point of failure in the fuel system.
- Five miles per hour (MPH) bumpers: the low-impact bumpers are forgiving of small mistakes at slow speeds made by the operator or other operators. Following an error (for example, automobile A moving at 4 MPH and striking with its front bumper stationary automobile B on its rear bumper), automobile A's bumper can absorb the shock of impact and automatically return to its preaccident front-end condition.
- Headlights that shut off automatically when the ignition is switched off: an inexpensive, simple relay. This feature prevents the fault of neglecting to switch off the lights from becoming the failure of a dead battery.
- An automatic airbag that deploys from the operator's steering wheel during a severe accident: in the split second between the impact of the automobile and the time when the operator's head strikes the airbag, the automobile's sensor has detected the error condition (the extreme deceleration).
- Seat belts for the occupants: can often prevent severe injury during an accident.

Under failure recovery:

- An auxiliary fuel tank with manual switch-over: the automobile's operator can use a backup fuel system to be activated manually in the event that the primary fuel tank runs dry.
- An auxiliary (backup) battery with manual switch-over: similarly, if the primary battery's power is drained, the operator's contingency plan is to activate the secondary battery unit.
- A cellular telephone or a citizens' band (CB) radio: if the automobile breaks down, the operator is able to call for assistance.

The underlying assumption of these strategies is that systems (or components) that are *not* broken can be depended upon, unlike those that *are* broken. Also, a dependable computing system's lifetime can be separated into two phases:

- Construction phase

  The period of planning, implementation, training, and verification that precedes putting the computing system into production usage.
- Operational phase

  The period during which the system must justify all the investment that went into its construction by producing tangible returns.

Table 1–1 introduces an approach to using the primary strategies, during the construction phase and the operational phase, to enhance dependability.

Table 1–1 Primary Strategies to Enhance Dependability

Primary Strategy	Construction Tactic	Operational Tactic	Characteristic Enhanced
Fault prevention	Minimize defects	Minimize mistakes	Reliability
Error correction	Provide masking options	Contain impacts	Fault tolerance
Failure recovery	Provide remedial options	Repair components	Recoverability

1.4.2 Redundant Functional Units

Functional units are those components of a system that actually do part of the work. They may also be decorative or have some other intrinsic value but for the purposes of this analysis, their function in the whole computing system is what is important. When there are multiple units that can perform the same function, the system has **redundant** functional units.

There are many types of redundancy. To understand this observation, the remainder of this section continues with the automobile metaphor introduced in Section 1.4.1.

1.4.2.1 Manual Redundancy

In Figure 1–3, the automobile's rear storage compartment holds a spare tire. This component's sole purpose is to provide a measure of redundancy in the event of a tire failure, to allow the driver to continue on the journey.

Such events are sufficiently rare that requiring the driver to manually replace the failed tire with the spare (or call a service person to do so) is considered an acceptable recovery strategy.

1.4.2.2 Automatic Redundancy

In Figure 1–3, there are two devices that represent key components of the brake system; they are the master cylinders. When the brake pedal is depressed, it is the master cylinder that causes the brake mechanisms on each wheel to engage and to slow the vehicle. In all modern automobiles, depressing the brake pedal actually engages two master cylinders, each of which is responsible for stopping two of the automobile's four wheels. The vehicle has two independent braking units.

In an emergency situation, there is clearly no opportunity to conduct a manual repair of the brake system to enable it to stop the automobile safely. If the braking unit associated with one of the master cylinders fails, the other braking unit continues to function with no action required from the driver and with no interruption in service. Therefore, automobiles with dual master cylinder braking systems could be described as having *fault tolerant* brakes.

1.4.2.3 Capacity-Related Redundancy

The automobile in Figure 1–3 is equipped with a four cylinder engine. The number of cylinders was probably chosen based on power, size, and weight requirements of the vehicle. However, because the number of cylinders in the engine is more than one, there exists a measure of redundancy as well.

If one of the four spark plugs in the engine fails for some reason, its cylinder would no longer provide power to the vehicle. It is likely, though, that the driver could continue to drive to a service facility because the other three cylinders would continue to motivate the automobile. The passengers may not be very happy with their rate of progress up hills, but at least they are not stranded.

A similar type of capacity-related redundancy exists with headlights at the front of the vehicle. If one headlight burned out, the driver could still have some illumination by which to return home. However, because there are only two headlights in the system, sustained operation at one-half illumination could be considered irresponsible and the driver could properly be cited by a traffic officer.

1.4.2.4 Redundancy in Computing Systems

These concepts apply equally well to computing systems. A computer's central processing unit (CPU) is frequently referred to as the engine of the system, because is provides the driving force for the other components. Later in this handbook, you see how computing systems with multiple CPUs have a measure of capacity-related redundancy. Keeping a spare video terminal in a supply room is analogous to having a spare tire, and mirroring your system disk with *volume shadowing* is equivalent to the dual master cylinder brake system.

Redundancy does not apply only to hardware components. Just as someone needing to drive a very long distance without stopping for sleep could take along a friend to help drive, a line printer operator could be cross-trained to handle disks and tapes as well.

1.5 How a Dependable System Supports the Business

The following list identifies the six basic building blocks of dependable computing systems and presents typical examples of each:

- Environment

  Physical security, power supply, and air conditioning.

- Hardware

  Computing machines delivered by the vendor(s).

- Communications

  Data transmission over networks.

- Software

  The programs that tell the hardware what to do.[1]

- Operational procedures

  The policies set by the information systems manager and the system managers that govern how people and applications may use the computing resources.

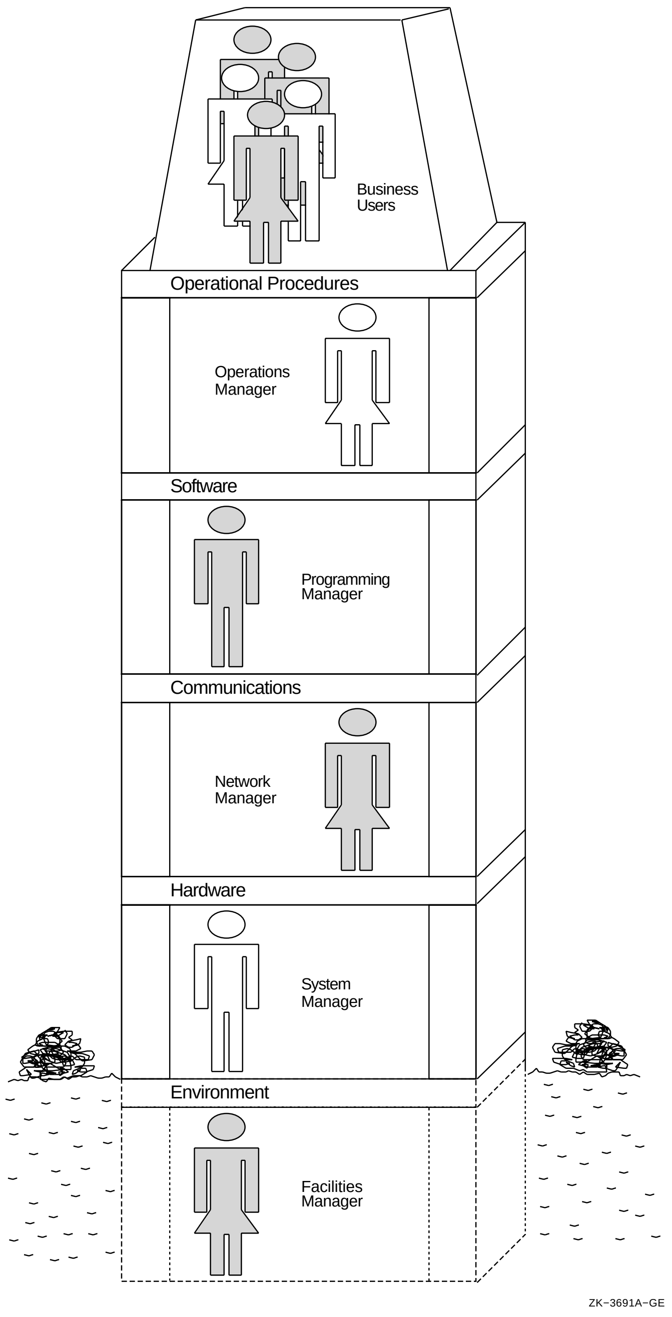
- Personnel

[1] Software always resides on a physical medium, which is a type of hardware device. The medium is not the software, the arrangement of bits is the software.

The people associated with the computing system, both vendor and customer employed.

Figure 1–4 uses the metaphor of a multilevel building to illustrate the building blocks of dependable systems, including the personnel who perform dependability tasks.

Figure 1–4 Building Blocks of Dependable Systems



In Figure 1–4:

- The environment building block forms the foundation of the dependable system, without which nothing else could function. The facilities manager is the person responsible for providing a satisfactory environment. That person uses both strategies for enhancing reliability, like using uninterruptible power systems (UPS), and strategies for enabling recovery, like preparing a backup site.

- The system manager, in this example, owns the responsibility of keeping the hardware operational. If reliability is paramount, the system manager could use fault tolerant hardware and hardware recovery techniques such as volume shadowing.
- Communications has a multidisciplinary nature and fits between hardware and software. So it is up to the network manager to make sure that the system's users (clients) have dependable access to all system resources and that all major parts of the overall enterprise system can communicate effectively with the other parts. A good reliability strategy would be the use of shielded twisted pair (STP) cabling to the desktops, instead of unshielded twisted pair (UTP) cabling, to reduce susceptibility to "noise." Because most communications software can adapt to line outages, configuring dual lines between key network points is an effective recovery strategy.
- The programming manager is, in this example, a combination of those persons who are responsible for ensuring that the software on the system (whether operating system, database, application run-time environment, or others) is functioning productively. Productively implies correctness with adequate performance. By reusing proven code modules frequently, the programming manager enhances the reliability of the system's software. To enable the applications to recover from internal and external faults more effectively, a TP environment could be used.
- Operational procedures form the boundary between the technology of the computing system and the organizational culture of the business. The operations manager is responsible for ensuring that the computing system is suitable for meeting the enterprise's business needs. Even assuming that an adequate systems analysis was performed when the applications were designed, the operations manager must still work toward enhancing procedural reliability (through training) and certifying recovery procedures (through drills).
- Clearly, both customer and vendor personnel are integral to the success of any dependable computing system. They have significant impact at all levels. If focusing solely on the technology permits the human factor to be ignored, reliable performance and readiness of personnel for critical recovery are likely impaired.

Figure 1–4 shows the interdependent nature of the dependability building blocks. It is also clear that an appropriate balance between reliability and recovery strategies is necessary for a dependable system.

The three primary dependability strategies outlined in Section 1.4.1 can be applied to each of the six basic building blocks. The form of that application is different for each building block and also depends upon the phase (construction or operational). Table 1–2 gives some examples of common techniques and practices.

Table 1–2 Applying Primary Dependability Strategies to Building Blocks

Primary Strategy	Building Block	Typical Construction Tactic	Useful Operational Tactic
Fault prevention	Environment	Excess/redundant capacities	Periodic maintenance
	Hardware	Simulation, CAD/CAM, parts count reduction	Operation well within specifications
	Communications	Robust shielding	Professional cable installation
	Software	Structured or object-oriented methods, CASE, prototyping	Live data parallel testing
	Operational procedures	Early and plentiful user input	Clear lines of interpersonal and intergroup communication
	Personnel	Comprehensive training, easy-to-use human interfaces	On-the-job training and practice
Error correction	Environment	Battery backup, high temperature shutdown	Power conditioners, uninterruptible power systems (UPS)
	Hardware	Error correcting codes, redundant recording	Instruction retries, micro head positioning
	Communications	Retransmit protocols, stack unwinding	Failing over to backup communication adapters
	Software	Retransmit protocols, stack unwinding	Input editing, handling hardware faults
	Operational procedures	Designing in constant feedback to operators	Supervisory and peer monitoring, giving praise for disclosing and fixing mistakes
	Personnel	Hands-on training, forgiving human interfaces	Supervisory and peer monitoring, giving praise for disclosing and fixing mistakes
Failure recovery	Environment	Media fire storage, Halon® fire control	Backup sites
	Hardware	Redundant zones, VMScluster systems, shadow sets	Automatic failover, manual repair, system replacement
	Communications	Automatic reconfiguration of network topology	Automatic reconfiguration of network topology
	Software	Powerfail-restart support	Dependable transaction queues
	Operational procedures	Contingency plans	Drills, updated plans
	Personnel	Contingency plans	Drills, updated plans

1.6 Balance Is Critical to a Dependable System

This handbook discusses the various building blocks, introduced in Section 1.5, that comprise a dependable system: environment, hardware, communications, software, operational procedures, and personnel. While this breakdown facilitates analysis, you should keep in mind that each of these major system components has a number of intrinsic and behavioral characteristics. They include:

- The performance envelope
- Initial and on-going costs
- Intrinsic reliability
- Security properties
- Ease of learning
- Ease of using
- Ease of maintaining
- Resilience against environmental hazards
- Suitability in addressing business requirements

What constitutes a proper balance of these characteristics depends on your organization’s dependability and business goals. These goals form the unique context in which your system must function effectively and dependably. One of the goals of this handbook is to help you analyze your situation and synthesize a credible strategy for addressing those sometimes contradictory goals.

2 Analyzing Dependable System Requirements

Because the system users are the ultimate judges of whether the computing system provides an adequate level of dependability, it is not realistic or useful to attempt to analyze your dependability requirements purely by means of charts and statistics. For example, achieving 99.99% up time has no practical value if the 0.01% amount of time that the system is down causes users to complain that the computer is down again. When this is the case, you can safely assume that there is a significant mismatch between the dependability needs of the business and the level of dependability provided by the system.

Although building a highly dependable system should not be a higher priority than the primary business needs of your organization, it is important to remember that the expectations of the users of the system are also a part of your system requirements. The purpose of the computing system is, after all, to help solve your business problems.

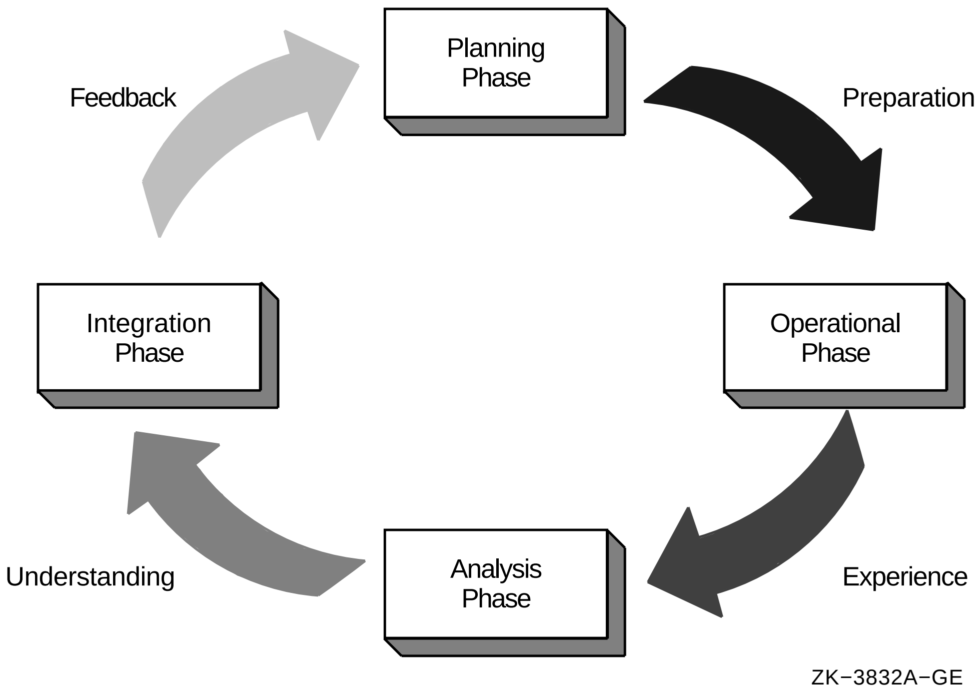
As you succeed in providing a certain level of dependability, you might discover that your users take the higher level of dependability for granted and come to expect an even higher level. Building a system that is dependable enough to meet the needs of its users is often a moving target.

2.1 Dependability Is a Journey, Not a Destination

You can maintain an appropriate perspective on the job of building a dependable system if you remember that dependability is a continuous task. The job of attaining a dependable computing system is never finished because you must regularly analyze your dependability needs and adjust your computing environment to respond to the changes in your business environment and practices. Constant vigilance is required and continual adjustments may be required.

To fully realize your system's dependability potential, you should implement a continuous improvement process, as illustrated in Figure 2–1.

Figure 2–1 Continuous Improvement Process



The quality engineering discipline uses a process called the **continuous improvement process** that provides structure to the task of improving how a process is executed. That process could be a manufacturing process, a business financial process, or other types of processes. This process has been adapted for application to the process of building and running computing systems. The following list summarizes this methodology for implementing a process to continuously improve the dependability of your computing system:

- Planning phase

  This phase focuses on preparing for the future and attempts to answer "*What if...*" questions. In this phase, you should:

  - Research the dependability needs of your organization. See Section 2.2.
  - Identify the options available in the building blocks of your current system including the computing environment, hardware, communications, software, operational procedures, and personnel. See Chapter 3.
  - Analyze the various options and tradeoffs you could make to improve the system dependability characteristics. See Chapter 3 and Chapter 4.
  - Identify the constraints that limit your options and determine how you can work around the constraints. See Chapter 4.
  - Devise some concrete actions you can take to improve the situation. Match the impact of the actions with the priority dependability requirements of your organization. See Section 2.2.4 for an example analysis of the sample company.

  Chapter 6 through Chapter 10 describe actions you can take to provide a higher level of dependability in your hardware and network configurations, software applications and databases, and data center procedures.

As Figure 2–1 shows, the *preparation* from the planning phase is used as input to the operational phase.

- Operational phase

  This phase focuses on innovation in the current context and attempts to answer the question "*What is happening?*" In this phase, you should:

  – *Roll out* your planned dependability improvements and closely monitor their impact on the operation of your system. Many of the tools listed in Chapter 9 are useful for these monitoring and control activities.

  – Keep comprehensive records of events (paper and online, as well as manual and automated) that have impacted or could impact the dependability of your system. These records will prove to be invaluable in the analysis phase.

  – Listen carefully to your users. They are the ultimate judges of the success of your *improvements*.

Note

Including your users in the planning phase can prevent many complaints in the operational phase.

As Figure 2–1 shows, the *experience* from the operational phase is used as input to the analysis phase.

- Analysis phase

  This phase focuses on the past to discover cause and effect relationships and attempts to answer the question "*What did happen?*" In this phase, you should:

  – Invest the time to carefully study all the various records and reports that were produced during the operational phase. Chapter 10 provides information on the comprehensive systems expertise available from Digital that can help you interpret the data collected.

  – Sort the good news and the bad news into the six basic system building blocks (environment, hardware, communications, software, operational procedures, personnel) and brainstorm new improvements to enhance each area. Section 2.2.4 is an example of this type of effort.

  As Figure 2–1 shows, the *understanding* from the analysis phase is used as input to the integration phase.

- Integration phase

  This phase focuses on transforming learned lessons into tradition and attempts to answer the question "*What should happen?*" In this phase, you should:

  – Document the new setup or the new way of doing things in your *operational procedures* document. This should reduce your exposure to loss of the wisdom due to personnel changes.

  – Conduct brief postmortem reviews so that the process for significant enhancement projects gets reviewed after being implemented for a period of time. Then the empirical data that was reduced in the analysis phase can be combined with the personal observations on how well your organizational culture adapted to the changes. Perhaps they can be made less painfully next time.

– Distill the lessons learned into a set of concise requirements or constraints for the next planning phase. It is important to *close the loop* to ensure continuous improvement.

As Figure 2–1 shows, the *feedback* from the integration phase is used as input to the *next* planning phase.

The continuous improvement process should never stop. As the word *continuous* implies, you must go through the process repeatedly, until your system is perfect or completely constrained.

Further, the various pieces of your system will probably be in different phases at once. You may be planning a major upgrade to the storage portion (disk farm) of your system, while also analyzing the effects of a recent building power failure. You may be formalizing a new system upgrade policy while also monitoring the system's application performance in real time.

Finally, in order to effectively implement change, an organization must be ready to accept the consequences. Some examples:

- Proper wiring of the building may involve some troublesome construction.
- Sufficient hardware redundancy may involve operating with some excess capacity.
- Less noisy phone lines may involve greater lease rates.
- Using a robust transaction monitor may require some changes in the user interfaces.
- Tighter system security may require some people to relinquish their account privileges.
- Replacing a row of line printers with many small office laser printers should involve *proactive* retraining and career counseling for the printer operators.

See Chapter 4 for a more complete discussion of the tradeoffs involved in continuously improving your system's dependability.

See Appendix B for information about the book *Out of the Crisis*, which describes the continuous improvement process methodology.

Note

Continuous improvement means constant change so it is worth repeating: in order to effectively implement change, an organization must be ready to accept the consequences.

2.2 Determining Your Dependability Requirements

The first step in the dependability process is to research the dependability needs of your organization. The analysis process should prove useful even if there is only one user on the system because the analysis will reveal the various levels of dependability required by the business functions that rely on the computing system.

In general, not every user of the system has the same dependability requirements. Some users, such as hourly employees, might find that being able to access the system for 40 hours per week is adequate for them to perform their jobs. Others, such as the company president or the engineering department, might have good business reasons to require usage of the system at 3:00 a.m. on a holiday.

Similarly, not every user of a computing system has the same business priority. Keeping the system available on Saturday nights to enable programmers to work whenever they desire may not be as important as making sure the President of the company can read electronic mail at any hour of the day or night. Of course, there may also be times when the programming work should take precedence over the President's mail correspondence.

You must also consider how the different functions of the business might place dependability requirements on the system. Business functions such as electronic mail, order processing, inventory control, or manufacturing might have different constraints on the computing system when it is recovering from a fault. Some business functions, like batch processing, might not be affected by a momentary system pause as long as the recovery operation is automatic and there is no processing or data lost. Other business functions, like control systems that rely on continuous processing, might require that applications execute without stopping in order to prevent machine malfunction.

Not all recovery operations (such as replacing a printer ribbon) can be implemented in an automatic fashion. In some cases, human intervention is necessary before production operations can resume. If one of your critical business functions requires instantaneous recovery and is vulnerable to interruptions by a fault or failure that requires manual steps to correct the problem, a disaster situation might be imminent for your computing system.

The point here is to keep these various factors in mind while determining your business dependability requirements and analyzing your system dependability characteristics. Your goal should be to work toward providing as close a match as practical.

2.2.1 Collecting Requirements

Generally, you specify requirements in terms of the tasks that system users perform for different business functions. Requirements should include current and anticipated changes in business environment and practices.

Most users whom you interview will describe the tasks that the system must support only in terms of the way they currently perform those tasks. The users might not be aware of impending changes in business practices. Even if users are aware of an impending change, they may not be able to state in any detail how the change will affect the tasks performed for their business functions. Therefore, you should make a special effort to identify known changes that the business plans to make and keep those changes in mind when interviewing users. Sometimes, the need to accommodate certain business changes may be stated as a general goal for the system before you begin your work.

You can collect dependability requirements by creating and using the worksheet shown in Figure 2–2. The worksheet can help you record and format your user and business dependability needs in a way that facilitates analysis. You can construct a worksheet similar to the one illustrated in Figure 2–2 using a text editor or decision support tool (such as DECdecision™ software).

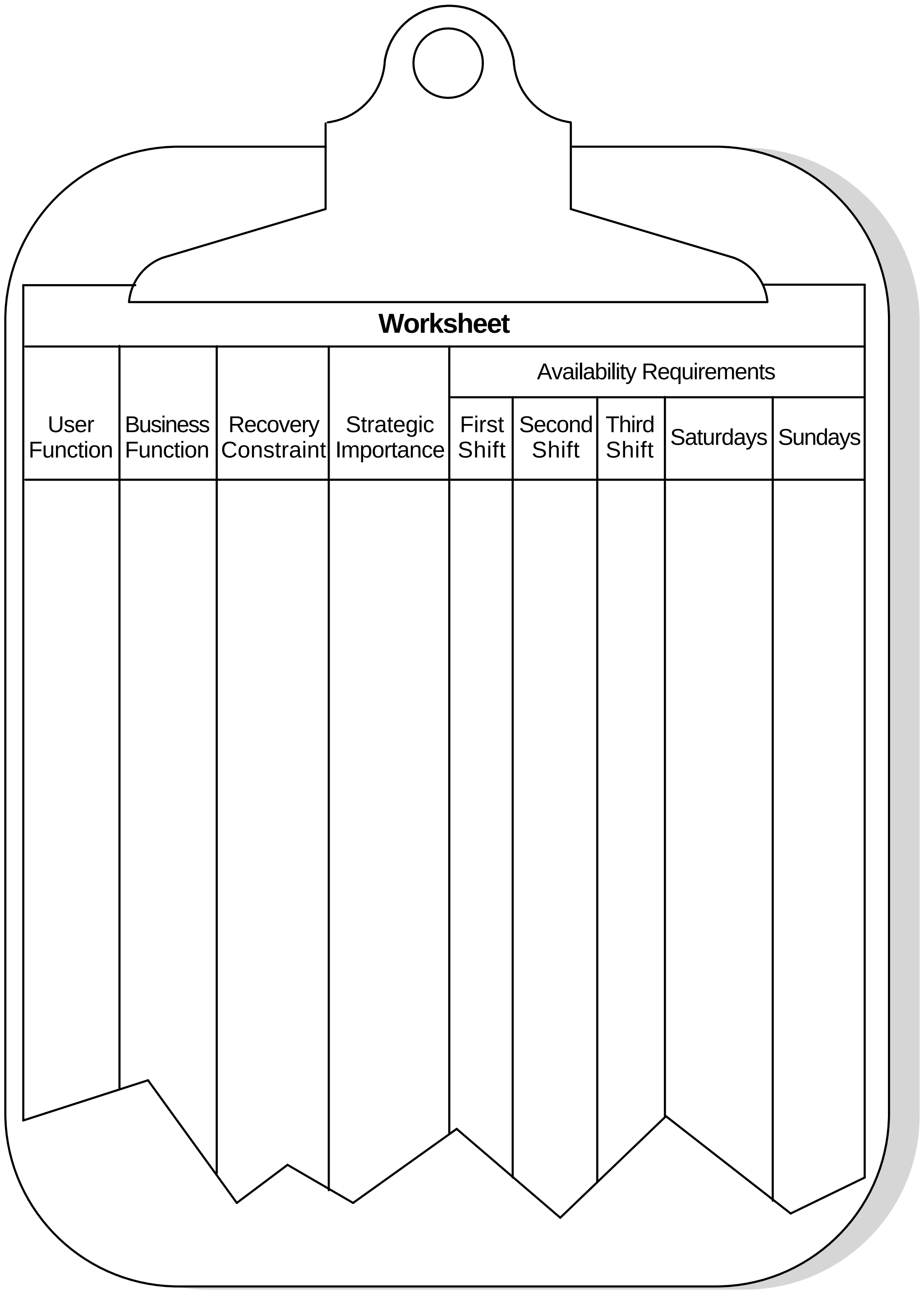


Figure 2–2 Worksheet for Collecting Dependability Requirements

Worksheet

				Availability Requirements				
User Function	Business Function	Recovery Constraint	Strategic Importance	First Shift	Second Shift	Third Shift	Saturdays	Sundays

The following list describes what information you should supply for the columns shown in Figure 2–2:

- User function

Supply the name of the user or the user's department. Examples are payroll, manufacturing, accounting, president, or the chief executive officer. Note that you will probably repeat some user and department names many times.

- Business function

  Describe the action or operation being performed by the computing system. Examples are electronic mail, inventory, time card processing, and simulations. Typically, the business function category describes the application program that accomplishes the task.

- Recovery constraint

  Assign an amount of time that would be acceptable for the recovery of the business function. Use the definitions of time in the following list to categorize the minimum acceptable recovery time:

  - Continuous or microsecond reaction time—The recovery of the business function must be invisible to both electronic and human interfaces.
  - Instant or subsecond reaction time—The recovery cannot be easily noticed by system users.
  - Fast or subminute reaction time—The recovery must be quicker than a person can place a phone call.
  - Slow or several-minute reaction time—As long as no processing or data is lost, the recovery time can be significant.
  - Manual or indeterminate reaction time—Automatic recovery is not required. Manual recovery actions are sufficient.

- Strategic importance

  The strategic importance column provides a way for you to express the dependability requirement of the entire business. The keywords high, medium, or low describe the degree to which the success of the business function determines the success of the business where:

  - High describes a very critical business function
  - Medium describes a business function of medium importance to the company
  - Low describes a low-priority business function

- Availability requirement

  The availability requirement column provides a way for you to express the dependability needs of individual business functions. List the time slots or shifts when your company uses the computing system (for example, first shift, second shift, third shift). As in the previous column, for each time slot, use the keywords high, medium, or low to describe how important it is for this business function to be available for use during that time. For example, printing payroll checks on Sunday morning would probably be considered a low availability requirement.

2.2.2 Priority Requirements

As you collect dependability requirements, the amount of data will grow and the values of the various attributes will probably change as well. The following list presents a useful approach to analyze and prioritize the data you collected in Section 2.2.1:

- Make sure that the information on the worksheet truly represents the major dependability needs and priorities of your company.
- Rearrange the rows in the Strategic Importance column in order from highest to lowest priority. If there are a large number of rows that have the same priority, you may need to use a finer-grained notation to indicate priorities. For example, you might use a scale from 1 to 10, where 1 is a very critical business function (high priority) and 10 is a very low priority.
- Sort the rows in the Recovery Constraints column in order from continuous recovery to manual recovery. Sort the recovery constraints within each strategic importance priority that you established in the previous step. For example, first sort the recovery constraint rows that have a strategic importance of 1, then sort only the rows with a strategic importance of 2, and so on. After this step, the worksheet will list your most critical dependability challenges first.
- At this point, you should examine the characteristics of your system's six basic building blocks to determine what your options are for achieving a certain level of dependability. See Chapter 3 for discussions on environmental, hardware, communications, software, operational procedures, and personnel options.
- Next, you must determine what trade-offs you need to make to achieve the desired level of dependability. Besides making trade-offs that affect global system characteristics, you may have to make trade-offs that impact some business functions negatively. This is necessary in some cases to provide adequately dependable service to business functions that are most critical to the success of the business. See Chapter 4 for a discussion of making trade-offs.
- Study your worksheet carefully to see if you can extract additional information from the data. By analyzing the data in various ways, you might discover that some users need their applications on Saturday but not Sunday, while others require their applications on Sunday but not Saturday. It is possible that some negotiation with the users could result in both sets of users requiring the computing system on Saturdays only, leaving Sundays free for scheduling hardware maintenance and software upgrades.
- Sort your worksheet rows again using Recovery Constraints as the primary key and Strategic Importance as the secondary key. At this point, your worksheet will list your most difficult dependability challenges first. It would then highlight which functions, if any, should require:
  - Conventional computing systems
  - Highly available VMScluster systems
  - Fault tolerant systems

Keeping your company's dependability requirements and priorities in mind, you can now begin to consider your options and weigh the trade-offs inherent with the different options. Section 2.2.3 describes how the sample company uses the requirements worksheet, and Section 2.2.4 proposes some steps to map the company's requirements to potential actions.

2.2.3 Analyzing the Sample Company's Dependability Requirements

Table 2–1 provides sample output of a fictional company's use of the dependability requirements worksheet.

Table 2–1 Dependability Requirements for the Sample Company

				Availability Requirement				
User Function	Business Function	Recovery Constraint	Strategic Importance	First Shift[1]	Second Shift[2]	Third Shift[3]	Saturdays	Sundays
President	Electronic mail	Instant	High	High	High	High	High	Medium
Payroll	Check printing	Manual	High	Low	High	Low	Low	Low
Sales	Order processing	Fast	High	High	High	Low	Low	Low
Manufacturing	Shock testing	Continuous	Medium	High	Low	Low	Low	Low
Engineering	Electronic mail	Instant	Medium	High	Medium	Low	Low	Low
Manufacturing	Inventory control	Fast	Medium	High	High	Low	Low	Low
Engineering	Simulations	Slow	Medium	Low	Low	High	High	High
Shipping	Packaging	Slow	Low	Low	High	High	Low	Low

[1]Primary business hours, 9:00 a.m. to 5:00 p.m.

[2]Secondary business hours, 5:00 p.m. to 1:00 a.m.

[3]Tertiary business hours, 1:00 a.m. to 9:00 a.m.

The following list describes each row in Table 2–1:

- President

  From the worksheet's first entry, you can see that this company places a high degree of importance on the president being able to communicate with subordinates who may be spread around the globe. Holidays and Sundays are the only times when there might not be a need to access the primary application, which is electronic mail. Because the president is an important system user, poor system responsiveness or noticeable pauses in character echoing or editing functions are perceived as roadblocks to productivity and are likely to generate high-level complaints.

- Payroll

  Similarly, the payroll department's ability to print checks during second shift is also a high priority. Notice, however, that recovery of the system while printing checks is not time-critical. The Recovery Constraint column shows that because the data center operations staff watches this batch operation very closely, it is not considered to be a problem to recover from system failures manually.

- Sales

  The sales department places a high priority on the order processing business function because this company has a reputation for being “easy to do business with.” Therefore, the order clerks must have constant access to a reasonably responsive system in order to respond to customer requests. A requirement for fast recovery time is reflected in the Recovery Constraint column. System pauses longer than one minute would be difficult to cover with casual phone conversation. Because order processing clerks must respond to toll-free phone calls from several time zones, they need dependable access to their applications during the first and second shifts.

- Manufacturing

  Manufacturing considers its shock testing to require a medium level of dependability. Because only a certain percentage of finished units are submitted to these tests, this sample company could continue to ship its product for some time even if the test facility was not in operation. However, when the shock testing is running, it is essential that the computer monitoring the shock testing equipment recovers instantaneously. The shock testing requires continuous recovery because even very short gaps in the responsiveness of the system (due to recovery or other actions) would cause test data to be lost. Because it costs the manufacturing organization considerable expense to run these particular tests, your analysis could prove that the shock test function would be a good candidate for a fully transparent, fault-tolerant system. Such a system would more than meet the availability requirements of the shock testing facility, which operates only during the first shift.

- Engineering

  Like the president, the engineers in this fictional company rely heavily on a dependable electronic mail function. Unlike the president, the engineers are not as likely to be accommodated if they demand exceptionally quick response time and recovery actions because the engineers’ ability to communicate using electronic mail is rated as only medium strategic importance. So, even though the analysis for the engineering department shows a requirement for instant recovery, the engineers might have to settle for fast recovery. Additionally, notice that the engineers’ requirements specify that they need a highly availability mail function during first shift, with their usage diminishing during second shift. This is because the engineers do not frequently communicate with the company’s international divisions.

- Manufacturing

  The manufacturing department also has a need for dependable computing for its inventory control function. While this function places greater demands on the computing system during the first and second shifts, the inventory control function does not have the same continuous processing requirement as the shock testing function. However, the production line would move too slowly if the system were not sufficiently responsive. So, the worksheet categorizes the inventory control application as having a medium level of strategic importance.

- Engineering

Engineering depends on the computing system for more than one major business function. To verify its product designs, the engineering department uses sophisticated computer simulation programs. These computational intensive batch jobs can run for several days, and quick recovery times are not required. It is important, however, that system failures do not force the jobs to be started over from the beginning. The ability to automatically restart applications is also highly desirable because most of the engineering simulation applications execute during the third shift and over the weekend to use excess computing capacity that would otherwise be idle.

- Shipping

  Finally, the system users in the shipping department who need the system during the second and third shifts may or may not receive the level of service they desire because the needs of some other departments might have a higher strategic priority than those in shipping. When business is slow, the pace of the shipping department becomes somewhat leisurely and occasional periods of system sluggishness are seen as cues for coffee breaks.

2.2.4 Generating the Sample Company's First Steps to Dependability

After studying the sample company's dependability requirements (shown in Table 2–1), a group of key members from the sample company's staff reproduced the worksheet shown in Figure 2–3 on a large board in a conference room.

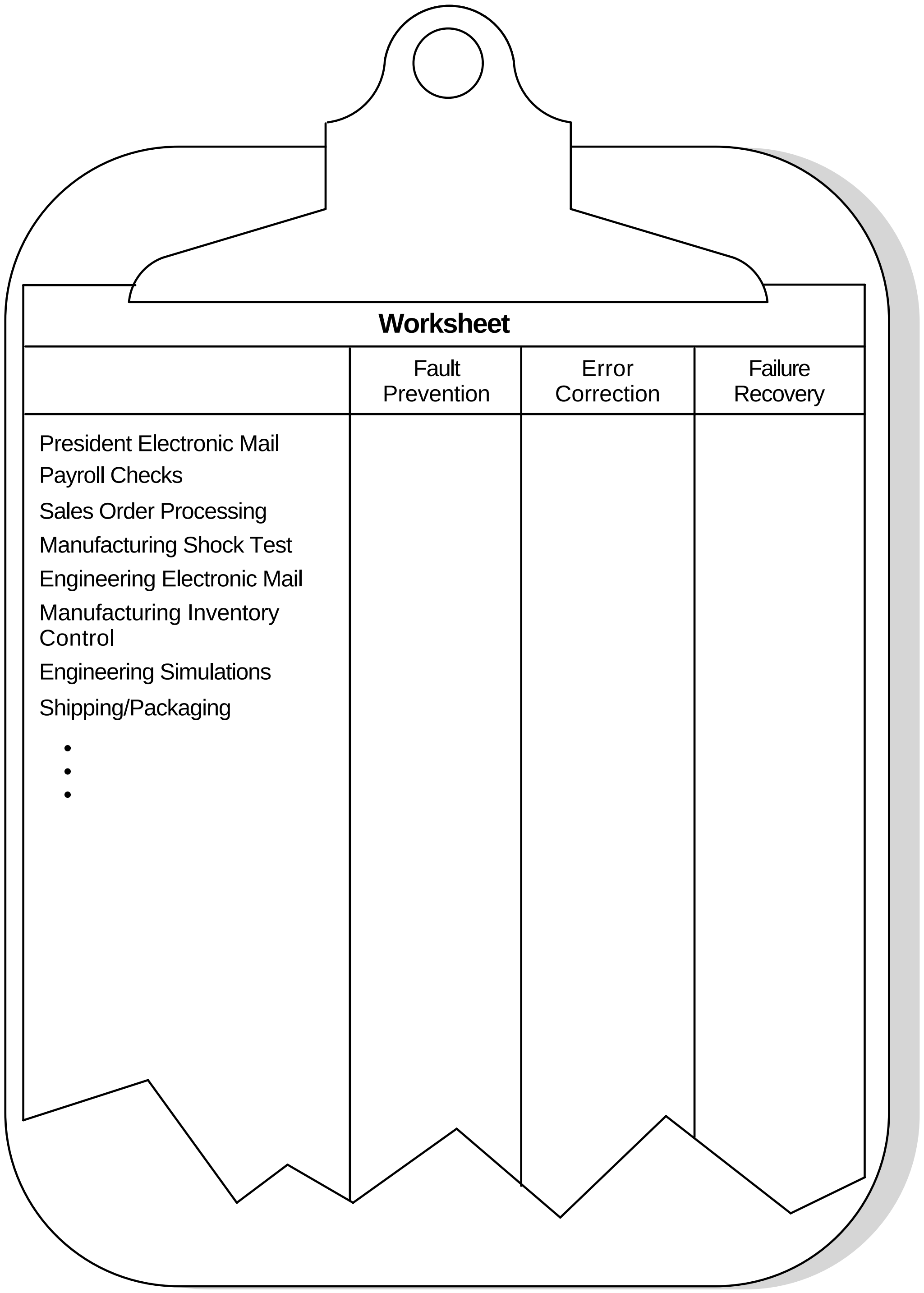
Based on the requirements determined in Section 2.2.3, the group brainstormed potential first steps to enhancing the dependability of their computing system. For each user of the system, the group members suggested ways to prevent faults that might occur for each user. Similarly, the group also recorded ways to enhance the error correction capabilities of their configuration and designed contingency plans for failure recovery.

The brainstorming session might have begun with concerns over the president's use of the system. Because the president's actions are considered critical, the group considered the use of both an uninterruptible power supply for fault prevention and shadowed user and system disks for fault tolerance. Although the group wanted to provide coverage by a system manager during second shift, the majority felt that the additional salary expense was not justifiable only to provide additional dependability for the president. However, the group was able to justify the expense of another system manager when they considered that the payroll department does their check printing during second shift.

While the group was on the topic of enhancing check printing, they considered the possibility of having the payroll and the president's data share a shadowed disk. They agreed that as long as the system manager set up the security on the user directories correctly, there should not be a problem with sharing the disk. In addition, the group stipulated that the shared disk must not be a shadowed system disk because performance monitor reports show that the system disk is already too busy and the additional load of printing payroll checks would drastically slow the system reponse time. Certainly the president would lose patience and consider the system undependable as a result of slow response time.

The discussion might proceed to provide a contingency plan in the event that the payroll department's printer would break. A viable backup plan might be to temporarily use one of the printers in the front office. The second shift system manager could set up this type of contingency plan for an emergency.

Figure 2–3 Worksheet for Brainstorming First Steps



Worksheet

	Fault Prevention	Error Correction	Failure Recovery
President Electronic Mail			
Payroll Checks			
Sales Order Processing			
Manufacturing Shock Test			
Engineering Electronic Mail			
Manufacturing Inventory Control			
Engineering Simulations			
Shipping/Packaging			
•			
•			
•			

ZK−3769A−GE

The discussions continued until the staff eventually filled the board with options. During the brainstorming session, members of the group were careful not to judge new ideas even though some options are radical and expensive. All ideas were recorded on the worksheet and later evaluated for technical feasibility and cost effectiveness. Some options were such simple, common sense ideas that the group wondered why they had not thought of them before.

Table 2–2 provides an example of the first steps that the sample company's staff proposed to begin enhancing the dependability of their computing system.

Table 2–2 The Sample Company's Proposed Enhancements

Proposed First Step	President Electronic Mail	Paycheck Printing	Shock Tests	Engineering Electronic Mail	Engineering Simulations	Shipping Packaging	Order Processing	Inventory Control
Install VAXft for shock tests			X					
Hire second shift system manager	X	X		X	X	X		
Buy TP-based inventory control								X
Give remote sites backup modems	X			X			X	
Checkpoint simulation programs					X			
Shadow president's disks	X							
Shadow all system disks	X	X	X	X	X	X	X	X
Write TP-based order process							X	
Add system kernel to VMScluster	X	X		X	X	X	X	X
Shadow payroll disks		X						
Install UPS in computer room	X	X		X	X	X	X	X
Train operators on TP system							X	X

2.2.5 Mapping Options to Requirements for the Sample Company

For each row under the column label Proposed First Step, an X has been placed in each user or department column in which the proposed first step would have a positive dependability impact. By examining the location and frequency of the Xs, the sample company can begin to sort through and analyze the dependability options:

- Rows that have many Xs indicate first steps that could have a wide impact.
- Rows that do not contain an X reveal first steps that would not impact any of the system users listed in the Table 2–2. These steps might still be necessary even though they are not important for dependability reasons.

- The columns of the most important system users should have several Xs in them. An important user whose dependability requirements are not being addressed by at least one first step probably needs more attention.
- When it is necessary to cut options from the list of requirements (for example, due to funding constraints) the sample company can search for rows in which the Xs impact users whose functions are less important. When the sample company must postpone enhancing the system dependability for some users functions, it delays those requirements for users and business functions that are not mission critical.

3 Dependability Options of the System Building Blocks

While an important principle of building a dependable system is the prevention of faults and failures, in your quest for a dependable system, you must:

- Assume some components of the system will fail at some time
- Ensure that when a failure occurs, it takes place at such a time or in such a way that it does not threaten service to the business

This chapter discusses options you could consider when looking for ways to enhance the overall dependability of your computing system. The discussions are organized within the main categories of the six basic building blocks of your computing system:

- Section 3.1, Analyzing Environmental Options
- Section 3.2, Analyzing Hardware Options
- Section 3.3, Analyzing Communications Options
- Section 3.4, Analyzing Software Options
- Section 3.5, Analyzing Operational Procedures Options
- Section 3.6, Analyzing Personnel Options

3.1 Analyzing Environmental Options

The computing system environment forms the foundation of a dependable system and it makes a good place to start examining your dependability options. Poor environmental conditions (such as high temperatures, humidity, or power irregularities) can lead to premature hardware failures and chronic intermittent faults that can cause the system to fail and not be immediately repairable. Reliable and clean power sources (or the use of an uninterruptible power system), freedom from environmental hazards such as weather, and security against intentional sabotage are all areas of concern.

The optimal physical environment for your computing system is best understood by your service vendor. The following sections discuss areas of environmental concern that you can consider for discussion with your service vendor.

These discussions provide an overview of your environmental options. See also Chapter 5 and Chapter 9 for more environmental considerations.

3.1.1 Utilities

Many of us take the reliable operation of our community's utility companies for granted. We never think about our assumption that the light will come on when we flip the switch. Further, when the light does not *come on*, it is usually due to a failed lamp or blown fuse; only rarely does the electric company deprive us of power.

To build a dependable computing system, you cannot afford to make such assumptions. It is not that your data center has much less reliable power than your residence (although such is certainly possible). It is that the consequences of even a momentary outage could be much more costly. Besides the computing work that could be lost and would have to be restarted, there may be damage to equipment controlled by your computing system or damage to the computer hardware itself.

Because poor quality power has been such a common cause of premature system failure, there are many products on the market that can effectively address that particular environmental concern. See Chapter 5 for more information about power conditioning units and uninterruptible power systems.

Air conditioning and water supplies must also be addressed with appropriate contingency planning. See Chapter 5.

Another utility company that could be critical to your operation is the telephone company (both local office and common carrier). If your computing system relies on telecommunications to reach its users or other important systems, a phone outage could render an otherwise perfectly functioning data center completely useless.

3.1.2 Structures

While the need to keep snow off laser printers is obvious, most computer equipment vendors have myriad other environmental requirements that are more subtle.

If your computer room is constructed on the third floor of a manufacturing plant, is there enough vibration in the floor to cause an abnormally high rate of disk drive (head crash) failures?

Although your computer facility may provide excellent environmental conditioning with redundant systems, if printer paper is usually carried in through the back door, will the tape drives receive gusts of hot humid air on a periodic basis?

Does your computer room have neat cable trays carrying all signal and power wires, or is it a place where it would be impossible to fix one piece of equipment without knocking loose the cables of another?

For overcrowded computer rooms, the good news is that newer equipment is usually much smaller and produces much less heat than older equipment of equivalent computing capacity.

3.1.3 Networks

Anything connected to your computing system can be an environmental hazard. Other computer systems may or may not be the source of intentional harm (such as security breakins or *viruses*). Even if they are not, they may cause your system problems due to programming or operational errors and misunderstandings.

The times when it is appropriate to isolate your computer system in the back room with the only communication being punched cards are becoming quite rare. Therefore, you must give serious thought to how your system interacts with some sort of network.

Are the communications programs robust enough to behave in a reasonable fashion, even if the other end of the line sends complete nonsense?

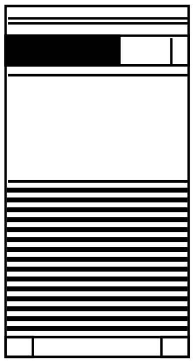
Are fiber-optic links used between buildings to provide electrical isolation and prevent electrical *noise* and possible shock hazards?

See Chapter 7 for more information about network concerns such as valid configurations and security.

Note

Your data center cannot function unless all the components of the environment are providing proper support. The environment sometimes hides single points of failure of the computing system. Because the rest of the system relies on the environment to function, an environmental failure impacts the entire system.

3.2 Analyzing Hardware Options



Because hardware components are becoming increasingly reliable, the set of hardware options available to you depends on the type of equipment you already own, the funds available to you for investing in additional or redundant hardware, or funds for investing in fault-tolerant hardware. Many companies have operated for relatively long periods of time (months or even years) without any problems, not only because of the reliability in modern computing hardware but also for a variety of other reasons. For example, some installations have exceptionally clean power sources and some have well-trained and exceptionally competent service people who can quickly diagnose and correct outages.

When you begin building a dependable computing system, you might be starting with new hardware or using older, already installed equipment. The following sections discuss the hardware options available to you whether or not you have funds to invest in dependable hardware equipment.

These discussions provide an overview of your hardware options. See also Chapter 9 for more considerations and dependability options you should examine to achieve high dependability in the hardware area.

3.2.1 Starting with New Equipment

Most dependability experts prefer to start out with new equipment if budgetary limitations or other constraints allow it. With new hardware, there is greater flexibility in configuring just the right solution for the business needs. In addition, new hardware is always more cost effective than older hardware.

From a dependability standpoint, configuring a solution from modern hardware is very attractive because new hardware is usually more reliable than older equipment. This is because new hardware has fewer parts and interconnections and requires less power consumption, which results in the equipment generating less heat. Modern hardware also includes more internal fault handling capabilities. All of these factors result in better dependability as well as lower service costs.

Often, the main obstacle to starting out with new equipment is budgetary constraints. You must evaluate whether or not the lack of a dependable system is costing you more than buying the new equipment that could help you achieve the required level of dependability. You have a better chance of meeting your dependability goals and configuring an optimal solution if you weigh dependability cost issues against your company's budgetary concerns.

3.2.2 Making Do with Currently Installed Equipment

If your currently installed hardware has not fully depreciated, there may be good budgetary reasons for not investing in new equipment. One advantage to making do with the hardware that you have is that you do not have to justify new capital expenses. You also do not have to wait for new hardware to arrive and to be installed. If you have not taken inventory of your current hardware, you should conduct a physical inventory to:

- List your current hardware
- Draw a map that shows exactly where each piece of equipment is located
- Show how the different hardware pieces are configured together

Taking inventory can point out irregularities in your hardware. You can use the configuration map to pinpoint areas in the total system where a mixture of old and new equipment might result in incompatibilities. If your records indicate that the older peripherals or CPUs are causing more than their share of down time and they are not contributing a large amount to the total system capacity, it may be more cost effective to retire the old hardware and replace certain pieces with new. The old equipment might be used for business applications that do not have high dependability requirements.

3.2.3 Investing a Little to Gain a Lot

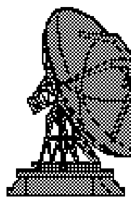
It is common for a data center to get approval to spend only small amounts of cash to enhance the system as long as the improvements are very noticeable. Fortunately, it is worthwhile performing an initial inventory and analysis of the system because in the analysis process you often uncover some high-impact, low-cost ways to enhance dependability.

Rather than trying to justify the cost of buying all new equipment or getting along with your current equipment, investing in a few new pieces of hardware could result in significant dependability gains for very little cost. To justify the cost of buying new hardware or software, it is useful to measure dependability characteristics in two ways:

- Show the amount of dependability you have in your present hardware configuration. Identify your current base level of dependability so that you can more easily spot the critical places where an addition to the configuration can make a significant impact.
- Show the amount of dependability that you can achieve with some improvements in the configuration. Present the dependability gains in terms of business impact that you expect to achieve by making the hardware enhancements.

Then, when you compare the current dependability of the system with potential gains through some configuration improvements, you can see more clearly what is to be gained from an investment in new hardware.

3.3 Analyzing Communications Options



The primary role of the communications hardware and software is to provide access from users' desktops over networks known as terminal networks to applications on the host node. Because host nodes usually support very large numbers of terminals, they are typically mainframe-class systems, like the DEC 10000 AXP™. VMScluster systems are also good candidates as AXP and VAX hosts to multiple users because of their high capacity and high availability features. The following sections provide an overview of your network and communications options. See also Chapter 7 for a discussion of considerations and dependability options you should examine to achieve high dependability in the area of communications.

3.3.1 Restricting Access to the Computer Room

You can reduce the computing system's vulnerability to computer viruses and other security threats by restricting communications to your computing system. By limiting access to all vital system components and keeping the computing system under the watchful eyes of experienced systems professionals, you can provide a highly dependable computing system.

3.3.2 Providing Corporate Data to Local Personal Computers

As more corporate desk tops are populated with personal computers (PCs), there is a growing trend toward linking personal computers together in networks called local area networks. These networks allow PC users to share spreadsheets and other common programs and data. If the local area network is also connected to the corporation's host node, the networks can greatly increase business productivity with increased access to corporate data.

The dependability of PCs can be enhanced by connecting them to a capable server system. The server system can allow data center and system management personnel the ability to provide remote control and system management support for the PCs. A big advantage to using a server system is that PC users can be relieved of performing system management responsibilities on their PC systems, such as data backup, software installation, and network control. Comprehensive server configurations, such as a Digital VAXserver™ 6000 running PATHWORKS™ software, also provide access to shared resources, such as laser printers. If a few small laser printers are conveniently located, the computing system can provide more dependable and functional hardcopy output to PC users than an inexpensive matrix printer on every desktop.

One key factor in providing dependable local area networks is in the type of communications medium that you use. While shielded twisted pair (STP) cabling has higher reliability than unshielded twisted pair (UTP) cabling, the latter can be qualified with special equipment used by your network vendor. Of course, coaxial cabling (such as Ethernet) or optical cabling provides even greater signal reliability.

Note

Have experienced professionals install and qualify computer cabling. Electricians who have not been trained about the installation constraints of high-bandwidth computer cabling can cause subtle network problems later on.

3.3.3 Weaving a Tapestry of Computing Resources

If you decide to connect your widely dispersed computing systems into a wide area network, you gain some disaster tolerance because the nodes on the network can be geographically distant. Clearly, a network spanning a continent is not very likely to have all of its systems disabled by the same natural disaster. However, to enable your users to continue to use their computerized business functions, the surviving systems must be able to service user needs.

To help ensure that critical applications and current data are available on multiple nodes in the wide area network, you can use:

- VAX Remote System Manager (VAXrsm) software, also referred to as VAXrsm, which ensures that remote systems have software to match the software installed at the central sites.
- DEC Data Distributor software, also referred to as Data Distributor, which replicates DEC Rdb™ relational databases. By making controlled copies of data in databases, the data distributor can be used to ensure that users of the network have access to the data, even if the central corporate site (or the network itself) fails.

Careful planning is necessary, of course, to be sure that the timeliness of the replicated data is suitable for the business functions using it.

The primary dependability constraints in a wide area network are the physical links. Because wide area network links typically have much lower reliability than local area network links, the importance of network topology is critical. In wide area networks, providing redundancy means having multiple physical connections between the various sites. Redundancy may be as simple as connecting the sites in a ring and maintaining connectivity by letting pass-through routing adapt for any single link failure. Redundancy can also be as complex as interconnected rings and trees that use multiple link technologies from several different common carriers. The impact of technology, bandwidth, and cost on physical link technology will be discussed in more detail in Chapter 7.

Many times, network performance both before and after failures dictates the design of the network. Refer to Chapter 10 for information on network design services.

3.3.4 Providing a System Network

As network technology improves, wide area networks are also improving so that they are starting to gain many of the throughput and reliability characteristics of local area networks. As a result, system and network managers are able to put fully distributed applications into general production.

Fully distributed applications can often support your business in many new ways but they may also make your business vulnerable to down time in other ways. When implementing fully distributed application systems, you should ask yourself the following questions to make sure the solution is sufficiently robust:

- How redundant is the underlying network topology?
- What happens to the performance characteristics if a link or a node fails?
- Can the networking software adapt automatically and transparently?
- What assumptions does the application design make about multiple nodes being both available and reachable?
- What security precautions can be taken to ensure easy access to authorized network users and *no* access to unauthorized users?

Refer to Chapter 7 and Chapter 10 for more information about designing robust networked applications.

3.4 Analyzing Software Options

Because computer hardware has become more reliable and software has become increasingly more complex, software has become one of the major factors in determining the dependability of computing systems. The dependability of the software on your system can be complicated by whether:

- Your company has a programming staff that writes customized applications
- Your data center staff or your users are allowed to acquire software as packages from software vendors
- The system management tasks include supporting a large collection of various brands and versions of computer software

Typically, most companies choose a combination of software options. By reducing the number of types and combinations of hardware and software that you need to coordinate and to support, you may increase the dependability of your system software. If users of incompatible software are unwilling to migrate away from their favorite but obsolete software applications, you should explain the benefits to them of moving their applications. For example, you could mention the value of the data center staff being able to perform such system management functions as nightly remote backups, network distribution of updates, and more reliable access to the corporate databases.

The following sections examine the dependability of software that is custom written by your company's programmers or contract programmers and software packages that you purchase from outside vendors. These sections provide a high-level overview of your software options. See also Chapter 8 for a thorough discussion of considerations and dependability options to achieve high dependability in the area of software and applications.

3.4.1 Writing Custom Applications

A major advantage of writing your own custom applications is that you can build basic reliability into the code as you write applications to solve business problems. From a dependability standpoint, your main objective in writing applications should be to engineer basic reliability into the software so that it does not contain defects in the first place.

This section briefly lists several methodologies for writing reliable software. You can research these methodologies and others more extensively by reading other computer-industry literature. Refer to Appendix B, Bibliography for a list of some useful titles.

The following list provides suggestions for building dependable software:

- Robust application design

  Programmers should not only write application code that is free from internal defects, but also try to predict how users, networks, and other programs will use or access their application code. By designing the application to adapt to widely varying conditions and to behave reasonably even under extreme conditions, the application can continue to work and provide service to achieve your company's goals.

- Computer-aided software engineering (CASE) tools

  There are many computer-aided software engineering (CASE) products on the market that can help the programming staff develop more reliable programs. By utilizing comprehensive modeling and analysis tools, software applications can be more suitable for solving your business problems and thereby improve the dependability characteristics of the computing system. In addition, the increased level of automation that you gain by constructing programs using CASE technology not only improves productivity, but greatly reduces programming errors. The Digital CASE tools are a part of the COHESION™ environment and include the following products:

  - DEC Language-Sensitive Editor/Source Code Analyzer (LSE/SCA)
  - DEC Performance and Coverage Analyzer (PCA)
  - DEC Test Manager
  - DEC Code Management System (CMS)
  - DEC Module Management System (MMS)

  In addition, the COHESION environment includes the CDD/Repository™ for its implementation of a distributed, active CASE repository. The repository allows the objects that are part of the software development life cycle to be managed and shared in a common, consistent fashion.

  See Section 8.7 for summaries about the COHESION environment products and pointers to the individual product's documentation sets.

- Fourth-and-fifth generation languages

Fourth-generation language (4GL) and fifth-generation language (5GL) programming technologies enable programmers to develop applications that are much easier to understand and maintain because the programmers do less work and the computer does more. The 4GL and 5GL technologies can make application software more robust by reducing the potential for application programmers to introduce errors. The following list provides a summary of many Digital 4GL and 5GL products. For complete information, refer to the specific product documentation.

- DECdecision

  DECdecision is workstation-based, information management and decision support tool composed of spreadsheet, database, and charting components. DECdecision has a format to help you store data, access and query data, analyze complex data, and create graphs based on that data.

- DEC RALLY™

  DEC RALLY is a menu-driven 4GL tool that allows you to develop prototypes and small-to-medium sized transaction processing systems quickly. DEC RALLY is tightly integrated with DEC Rdb, the Digital relational database system. DEC RALLY can also access RMS files.

- VAX ACMS™

  VAX ACMS, also referred to as ACMS, provides a 4GL task specification language with which programmers can write applications that accommodate many users as well as respond to changing business conditions in a timely, competitive manner. ACMS allows programmers to construct applications in modules that are easy to maintain and are easy to distribute to better meet the business needs.

- VAX OPS5™

  VAX OPS5 provides a very efficient language with which you can implement expert systems. The use of VAX OPS5 to apply artificial intelligence technology to production systems is being widely deployed within Digital and by Digital customers.

• Transaction processing environments

One effective way to construct very robust applications is to take advantage of a transaction processing (TP) environment. This decision has to be made when you are designing your applications because the application code must inform the TP environment when to begin, end, or abort transactions. Depending on the TP product that you choose, you can easily create highly available applications that will survive many classes of failures of your system, your VMScluster system, or your network. Digital provides the VAX ACMS (application control and management system) as its robust transaction processing environment. See the complete VAX ACMS documentation set for more information.

DECADMIRE™, which stands for Application Development: Making It Really Easy, is an application development tool that generates high-performance, large-scale production applications. You can use DECADMIRE to create full client/server and TP implementations across heterogeneous systems, standard timesharing applications or continuous fault tolerant applications, in the OpenVMS environment. Rapid development takes place using the prototyping, automatic screen generation, and code generation facilities. Applications developers can use DECADMIRE to describe and define the database, and manage the appearance of application screens, menus, and reports. DECADMIRE combines applications development packages for:

  - ACMS, COBOL, and DECforms™
  - ACMS, Pascal, and DECforms
  - ACMS, FORTRAN, and DECforms

- CASE tools for adequate testing

  Companies that have succeeded in operating extremely dependable computing systems know that the best way to ensure that a software module functions correctly in a production environment is to put it through extensive testing. Testing should involve supplying the same data to the test system that you are supplying to the production system and then comparing the results. When both the test and production systems produce the same results, you attain a high level of confidence that putting the new application into production will have minimum risk. In addition, it is wise to supply data to the test system that tests its error detection and recovery capabilities to ensure that a worst-case scenario does not render the application useless. See Section 8.7 for information about the Digital COHESION environment products that can assist testing efforts. Also see Section 8.4 for information about prototyping applications and Section 8.5 for information about testing methodologies.

- Comprehensive training

  Even with the best tools available, your programming staff cannot succeed unless you also provide sufficient training. If you cannot significantly enhance the tools available in the development environment, you can usually make your programmers more effective with their current tools by increasing their level of competency with them.

  In addition, do not overlook adequate training for the users of your computing system. The better trained the users are, the less likely they are to make mistakes when using your system. See Chapter 10 for more information on training programs and resources.

- Quality metrics and mindsets

  Mature (and therefore highly reliable) development processes establish a way of measuring the quality of the software. When you measure the effectiveness of the application software, you automatically facilitate improvements to the software itself. If you are able to reduce the number of times in a month that an application needs to have software errors fixed, you probably also reduce the number of times that the application is not available to system users. Similarly, applications that work successfully also help reinforce good programming practices and help your programming staff develop the mindset to do it right the first time.

3.4.2 Acquiring Software Packages

Purchasing software packages rather than writing your own software has the advantage of requiring far less programming staff and time to implement than building your own applications. If you can find a software package that is suitable to your needs and that is also affordable, purchasing ready-made applications can be a very effective way of solving a wide range of business problems. The following list describes several factors you need to consider before you acquire software packages:

- Degree of fit

Most software packages allow you some ability to customize the application to fit your business needs. However, be aware that if the basic characteristics of the software package are very different from your organization's way of doing business, your system dependability may suffer. A software package that does not match the system user's style of working may indirectly cause an abnormally high level of errors and mistakes. Because people are part of the system, you must pay as much attention to their reliability as to the reliability of the other system components.

- Service and maintenance arrangements

  If a vendor's software package breaks, the good news is that you do not have to fix it. The bad news is that you usually have to wait for the vendor to fix it. Responsiveness of the service team could become a critical factor once your company has grown to depend upon the package's functions.

  Besides service policies, you should give consideration to the software maintenance practices of the vendor. How often are problem-correction releases sent proactively to customer sites? What is the scope and duration of the job to update the package to the next revision? Must the vendor's staff perform the upgrade or can your own staff do it at the time most convenient for your company's operations?

- Vendor reputation

  If your company is holding you responsible for the dependability of its computing systems, give yourself every chance to succeed by obtaining the highest quality products and services you can find. Your own reputation is riding on your vendor's reputation.

- Reference sites and support groups

  Before acquiring the software package, pay a visit to other companies like your own who are also utilizing it. This can help you determine if the package would actually fit your company's needs as well as it should. It is also an excellent way to acquire first-hand information about a vendor's *real* reputation, as seen by its clients.

  Locally accessible support groups for your prospective software package are important because any complex set of programs can behave in subtle and surprising ways. Many times, the staffs of those other sites can be a valuable source of operational experience and wisdom, which can help you proactively avoid problems in your own system.

3.4.3 Selecting a Systems Integrator

The main coordinator of a complex project such as building an office building or building a dependable computing system is sometimes referred to as a "prime contractor" or a "systems integrator."

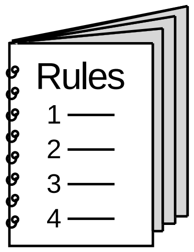
The job of a systems integrator has many characteristics similar to those described in Section 3.4.2 and has most of the same considerations that need to be addressed. In a prime contractor role, however, a systems integrator would probably deal with multiple vendors and many organizations within your company.

One advantage of this approach is having only *one* party responsible for the success of the implementation. Again, a vendor with a solid reputation for quality is a key factor in providing a dependable computing system for your users.

An alternative option to hiring a systems integrator is hiring a number of contract programmers to augment your own staff. Utilizing contract programmers is very similar to the factors listed in Section 3.4.1 without the overhead of additional permanent employees.

Refer to Chapter 10 for information on how Digital can assume the role of systems integrator for your computing system.

3.5 Analyzing Operational Procedures Options



This discussion provides an overview of your data center and operational procedures. The following sections describe some approaches including training, automation of operation procedures, beepers, and overall business policies that can help system users become more dependable contributors to the total system. See also Chapter 9 for a thorough discussion of considerations and dependability options to achieve high dependability in your operational procedures.

3.5.1 Avoiding User Errors

The people who interact with the computing system are an integral part of the entire system operation. As such, the dependable operation of your system rests as much with the capabilities of your personnel as it does with a dependable power supply. Personnel who are trained to be competent system users can make a positive impact on the dependability of the computing system. Your goal should be to minimize the number of mistakes made by your personnel because user errors are a very significant contributor to system and application down time.

The best way to avoid user errors is to automate operational procedures to make the computing system do more and the people do less. You can minimize user involvement by automating as many of the system monitoring and management operational procedures as possible. When the system requires user input or intervention, simple and concise system messages should provide clear notification of the problem and should specify a recommended action whenever possible.

For those times when people must be involved in the operation of the system, you can make your personnel less error prone by raising their skill levels through adequate training. Dependable computerized systems can help you provide a return on your business investment. When computers are used in a competent manner, the process of building the competency of your personnel becomes an important topic for consideration.

3.5.2 Training, Testing, and Drills

The following sections outline some approaches to training your system users to be a more dependable part of your total system. See also Chapter 10 for more information about providing support and service to system users.

A dependable system user is one who has proper training to know what to do in any situation. Competent users know how to use the system and its applications to collect and analyze information, to make the right decisions, and to avoid mistakes and oversights. To achieve high competence in your system users and to reduce the occurrence of mistakes, you must emphasize training and practice.

Be sure to implement a comprehensive training program for users who are new to the system and also for all system users whenever you add new business applications to your system. Although your applications might have well-designed user interfaces that are tolerant of incorrect user input, even applications that have excellent user interfaces can fail if they are abused or misused.

Plan to train personnel running the data center to ensure that they are prepared to serve the system users in a reliable fashion. To this end, you should determine the minimum level of competency that your data center personnel should satisfy to meet your company's goals for mission critical computing.

Once trained and tested, the system users are still not totally prepared because the most important tasks your staff perform (from a dependability perspective) are required only in extraordinary situations. The system users do not have many chances to practice emergency recovery actions. Therefore, you need to provide artificial practice for those tasks via ongoing drills and evaluations.

3.5.3 Extended Hours of Operator Coverage

While one goal of lights out data centers (like the one described in Chapter 11) is to reduce costs and minimize the number of operators needed to support computing services, some businesses may not be able to migrate to a lights out environment anytime soon. If there is an immediate need for more hours of system usage, the business may have to invest in the staffing to provide the additional coverage. (It takes careful planning, aggressive management, and time to move to a lights out environment. See Chapter 11 for more information.)

If your users require the services of your computing system after normal working hours or even on weekends and if their applications require operator support, yet you are not ready or willing to pursue a lights out style of computing, you may have to schedule operational personnel to provide such support.

Some creativity may be required to work around limitations. For instance, if you are *fully staffed* with two 8-hour shifts of operators but the company needs 24-hour service and hiring another shift of operators is *out of the question*, perhaps instituting two 12-hour shifts (and paying some overtime) would be a way to make do with what you have.

3.5.4 Beepers for System Managers and Programmers

Many companies place their technical staff on call with beepers to ensure that, in an emergency, the system problems can be diagnosed and corrected as quickly as possible. If this represents additional responsibilities for your staff, consider how you are going to sell this new opportunity to them. If, for instance, you gave them a laptop PC with modem as well as a beeper, they would be able to start diagnostic and corrective actions sooner because dialing in is much quicker than driving in. Plus, they might be able to fix those month-end closing jobs on a Saturday night without having to leave their after hours activity.

3.5.5 Lights Out Computer Facilities

One way to reduce the likelihood of operator mistakes while increasing the physical security of your computer room is to turn out the lights and lock the doors. This requires that the normal production operation of your computing system can continue without the services of people frequently mounting magnetic tapes and disk packs or distributing printer output.

Because you would have a smaller staff of operators in such a situation, routine operational functions such as backing up data from disks to tapes would have to be made as automatic as possible. Peripherals such as stack-loading tape drives, which can handle many units of tape media (and therefore back up many units of disk media) without operator intervention are essential to such an environment.

The ability to detect remotely and to correct problems with the system becomes very important when there is too limited access to the computer facilities. Products such as the Digital VMScluster Console System (VCS) and Data Center Monitor (DCM) are useful in monitoring and controlling the hardware and software from a central location.

To help discern how your computing system can become free of constant operator attention, continually ask yourself the questions “What could the system (hardware, software, physical plant) do for itself?” and “What could my users (including my own staff) do for themselves?” Placing small laser printers in the office areas may eliminate the need for high speed, centrally located lineprinters. Utilizing software such as the DECscheduler™ product allows complex production runs to become computer managed. Many suppliers of environmental conditioning units offer options that enable remote monitoring from personal computers.

For more information, see Chapter 11 for the case history of how one of the Digital customer support centers was able to implement a lights out data center.

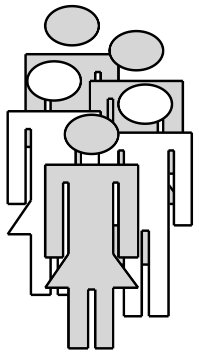
3.5.6 Policies Regarding System Privileges

Because the issue of system privileges involves the allocation of power to staff members, it is an area that requires significant management attention beforehand. Policies that start out too loosely are extremely difficult to tighten later, so your operation is best protected by starting with a rather restrictive policy and allowing exceptions on a case-by-case basis.

VMScluster systems present a special concern because their design goal is to have all the nodes function together as a *single system* with a single security domain. If your VMScluster environment is typical, there is a common authorization file for the entire environment. Therefore, if your programmers have system privileges on the development nodes, they will also have those same privileges on the production nodes. Such a situation may be necessary to allow the programmers to install corrected applications in a timely fashion.

It may be advisable in some cases to raise more of a barrier between the development environment and the production environment by giving the programmers their own network nodes or even their own VMScluster system. This arrangement increases the system management load on your staff, but it prevents the problems that stem from having staff members with system privileges using the production system for daily activities such as file and process deletion. See Chapter 9 for more information about system security.

3.6 Analyzing Personnel Options



You and your coworkers are an integral part of a computing system. Although this observation may seem obvious, experience has shown that some organizations overlook the importance of positive morale, comprehensive training, and a high degree of involvement among a data center's personnel as key factors for the data center's success. As illustrated in Chapter 1, there are people at all levels and in all areas of a system built with dependable building blocks. In that respect, you cannot separate the *human factor* from the other aspects of dependability because it is so closely intertwined with them all.

3.6.1 Teamwork Makes the System Work

Perhaps because we are so used to being around other people, we take them for granted. To ignore them in our dependability analysis would not be useful, however, because ultimately it is people who implement dependable systems and people who evaluate them.

Even though the facilities manager is shown in the basement of Figure 1–4, the role is not a minor one. By providing the foundation upon which to build a dependable system, the facilities manager shoulders an enormous responsibility for the enterprise.

There are also few thanks for the system manager who monitors the system and interfaces with the service vendor to "keep the hardware running." Most sites have a much broader definition of the system manager's role, but if the hardware (or operating system) breaks, that person takes the heat.

Straddling the boundary between the hardware and software worlds is the network manager (who is frequently also the system manager). That is because communication is so dependent upon both physical media characteristics and algorithmic protocol dynamics. Computers that can be completely isolated and still provide useful functions for the enterprise are a dying breed.

Of course, the programming manager is obligated to live at the front lines where the users are attempting to convert the computer's applications into productive work for the business. Like the people mentioned previously, this professional is dependent on the other computer systems personnel for support, and in turn, supports others that use the system.

Perhaps the toughest job of all is performed by the operations manager. One reason the operations manager's job is difficult is because the board of directors also likes to listen to the computing system's users. The board listens not only to the tellers or data entry clerks, but also to the managers of the business departments who have come to depend on the computer for survival. The real bosses, then, are the business users, upon whose success the enterprise depends.

3.6.2 Robots, We Are Not!

A key element of any dependable system is the comprehensive reporting of problems, whether they are hardware or software defects or operator mistakes. The promptness, completeness, and correctness of the data in a problem report is critical in determining what went wrong and how it might be avoided in the future.

Unfortunately, people do not like to admit their mistakes, especially if the mistakes reflect poorly on their job evaluation. This frailty is one of the contributing factors to a phenomenon that reliability researchers call, "the mystery reboot." When they analyze system logs to determine what caused the down time, they can isolate hardware-related crashes, privileged software induced crashes, and scheduled system shutdowns but there remains a large class of outages that show up only as "system rebooted." Without supporting data, the service vendor cannot determine what is broken and therefore does not know what to fix.

One possible way to alleviate this problem might be for management to respond to personnel who make excessive mistakes with additional training, rather than additional reprimands. While the dependability of a computing system is not helped by tolerating incompetence, an atmosphere of "professional safety" can promote the full disclosure of operational history that is so key to successful problem diagnosis.

4 Balancing Dependability with Other Business Considerations

Very likely, there are many factors limiting your freedom of action as you strive to provide a dependable computing resource. Before you can overcome these constraints, you need to analyze their nature, their causes, and their impact on the dependability of your system. This chapter addresses these factors.

4.1 Identifying Constraints to Achieving a Dependable System

To begin understanding the limits of your options for enhancing the dependability of your computing resource, write down replies to the questions in Table 4–1. Some of the questions may not apply to your business or systems. Additionally, the questions do not represent a complete enumeration of all possible constraints; their purpose is to help you start thinking about these issues.

Table 4–1 Dependability Constraints: Sample Questions

Constraint	Questions
Physical site	How much space do you have for your system? What are the power distribution and cooling capacities? How wide are the access doors, and how are they controlled?
Equipment funding	What is your budget for new equipment? What does it cost to maintain the old equipment? If the newer equipment costs less to maintain, could that increase the budget for new equipment? Is used equipment an acceptable solution to your needs? How would your company's finance department view leasing versus purchasing?
Software acquisition	What is your budget for software acquisition? What operating system and hardware environments must be supported? What levels (and localities) of support are required? Would it be possible to acquire special equipment for new software, if necessary? What type of software is already familiar to the current system users?
Program development	How complex of an application system could your personnel reasonably undertake? How would complexity impact maintenance efforts? How effectively could computer-aided software engineering (CASE) be introduced into the culture? What training would that require? Would it require any additional equipment acquisitions? What is the availability of outside contractors and specialists to assist your programming staff?
Staffing and training	Would it be possible to hire more system staff, such as operators and system managers? Must you reduce your staffing levels? What kind of training would help you get the most from your current staff? Are there any funds for that training? Are there alternative training options available, such as videotapes or computer-based instruction (CBI)?
Organizational	How does the culture and political structure of your company impact your efforts to provide a dependable computing resource for your users? What are the performance and dependability expectations of your most important users? Which senior managers could use the most education in the concepts and benefits of investing in dependable computing?

(continued on next page)

Table 4–1 (Cont.) Dependability Constraints: Sample Questions

Constraint	Questions
Legal and other external factors	What are the performance and compliance (to industry standards) constraints placed on your operation by government agencies? How does your company's competition influence the direction and rate of change of your system? Are there any health or safety considerations associated with the dependability of your system?

Do not become discouraged after you have listed the constraints you have to accommodate. The point of this exercise is not to convince you that you are in an impossible situation. Rather, the point is to help you gain a clear vision of what the boundaries of your path are, so you do not run into them as frequently.

As you analyze your computing system, you will be less likely to start planning a solution that you cannot implement because the boundaries of the solution space have been explored. Many times, what is technically feasible may not be completely achievable in the real world due to nontechnical constraints. Failure to consider all the factors that can constrain your path may result in a system that seems like it should work but is instead continually problematic.

An example of this would be a dependability enhancement that relied on a change in your company's operational procedures. If the computing resource's management and the system user's management were not *both* fully committed to implementing the change in their organizations, the enhancement would fail.

If your company has given you the responsibility of providing dependable computing services, you should understand and analyze the operational impacts of enhancing the dependability characteristics of your system. You can then help the general management of your company understand the business trade-offs involved.

The following sections discuss these business considerations and their relationship to dependability. Keep the following maxims in mind while you evaluate your options:

- Building a dependable computing resource is not an end in itself.
- "Dependable enough" is determined by the business requirements you (continuously) are attempting to meet.

4.1.1 Performance Tradeoffs

Because no physically implementable system is perfectly reliable, a dependable system must consist of more than one independently recoverable unit (IRU). This allows the system to continue to provide service, even if one unit fails. See Chapter 6 for more information on how total system performance can vary over time with IRUs. There are several ways to design IRUs. This section describes the performance characteristics of IRUs.

After the hardware-related redundancy issues are covered, the impacts of dependability concerns on software performance and staff productivity are addressed.

4.1.1.1 Circuit Level Redundancy

Many large CPUs, like the DEC 10000 AXP and the VAX 10000, have comprehensive internal consistency checks and sophisticated error correction hardware and firmware. The DEC 10000 AXP and VAX 10000 console processors monitor and control their continuous self-checking actions. Consult the CPU systems documentation for details.

The performance impact of circuit-level redundancy is usually not measurable. Unless really serious problems occur, such as having to disable the CPU's high speed cache memory, application programs (or your system's users) cannot detect the operation of these recovery mechanisms.

4.1.1.2 Subsystem Level Redundancy

Many systems from Digital, such as the DEC 7000 AXP and the VAX 7000, can be configured with more than one CPU per system kernel. The CPUs share a common main memory and each CPU performs equivalent processing functions. These multiple CPU configurations are known as symmetric multiprocessing (SMP) systems.

The OpenVMS operating system has the ability to continue operating in many situations where one of the CPUs in an SMP system has failed, but at least one remaining CPU is still functioning. This built-in redundancy of a key subsystem is a major advantage of SMP configurations. Even in situations where a CPU failure causes a system crash, an automatic reboot operation can restore service more quickly than an on-site service call.

However, with one CPU out of operation, the resulting system has less processing capacity than before. For this reason, you may want to consider configuring an SMP node with one more CPU than is minimally required to support the mission-critical workload. Then, if a CPU fails, your system can still achieve its performance requirements. This class of redundancy could be called (n+1)[1] because the approach is to acquire one more unit than is nominally required: to retain nominal behavior in the presence of a single failure.

OpenVMS can also treat two or more disk drives as redundant storage devices. This capability is provided by Volume Shadowing for OpenVMS and can make almost any disk in a VMScluster system a redundant twin of any other same-model disk anywhere in the same VMScluster. This class of redundancy could be called (2*n) or (3*n) because this approach involves installing multiple units of the disks to be shadowed. Because of the expense, many companies use volume shadowing only on their mission-critical and system-critical disks.

Because data writes to disk must be sent to two (or more) devices, there can be a slight performance degradation for write operations compared to nonshadowed disks. There also can be a slight performance improvement for read operations because one disk may locate the data before the others.

You must also consider the time domain behavior of shadowed disks during recovery operations. If one of the shadow set members fails and is subsequently replaced with a brand new disk, the operating system must make the new disk's contents exactly match the other shadow set members. This, of course, involves overhead I/O operations that place additional loading on the disk controllers, interconnects, and the access arms themselves.

See the *Volume Shadowing for OpenVMS* for more information.

[1] Where "n" is the number of objects.

4.1.1.3 System Kernel Level Redundancy

A **system kernel** is the combination of CPU, memory, and I/O port. System kernel redundancy is different from SMP configurations because in an SMP cabinet, the multiple CPUs share a common memory subsystem and a common set of I/O ports. Therefore, an SMP node counts as a single system kernel.

The following list describes several approaches to creating redundancy at the system kernel level, each with different performance characteristics. Note that the following discussions assume that the issue of dependable data storage is addressed separately; the following list deals with providing multiple compute engines:

- Tightly coupled system kernels

  One way to ensure that there is no pause in service from the computing resource, even when a major subsystem fails, is to dedicate more than one system kernel to the same task. By keeping a primary and a secondary kernel in *lockstep*[1] as the kernels process instructions, the failover transition can take place quickly. The primary performance advantage of this approach is that *continuous* processing is possible. A hidden performance disadvantage of this approach is that while multiple CPUs are required to implement this configuration, the application executes only at the speed of one of the CPUs (2*n redundancy). The others are used in the failure detection and recovery operations, not for additional application performance (as would be the case with a decomposed application executing on an SMP system). See Chapter 6 for more information on the Digital VAXft series of fault tolerant systems, which use the tightly coupled approach.

- Active-standby system kernels

  Another approach that has similar performance characteristics as tightly coupled is the active-standby configuration. In this case, two completely separate system kernels are dedicated to one purpose, but the CPUs are *not* kept in lockstep. There is a short pause when the active kernel fails and the standby kernel has to take over the application processing. Besides the short failover delay (which varies according to processor), it is still the case that more than one kernel is dedicated to one set of applications with no increase in application performance (2*n redundancy). One advantage of this approach is that because the CPUs are not kept in lockstep, software defects that cause the active kernel to fail do not cause the standby kernel to fail simultaneously. See Chapter 6 for more information on the Digital MIRA AS systems, which use the active-standby approach.

- Loosely coupled system kernels

  With loosely coupled system kernels, which access common data storage, you can configure a dependable system that can make full use of all the kernels all the time. The performance characteristics after a system kernel fails depends on your configuration choices. If you have funding (or space) for only as many system kernels as required to provide adequate application performance, your computing resource will have to operate in a degraded mode if a system kernel fails. Depending upon your business context, you might have to let your system users interact with a less responsive system or you might be able to shut down less critical applications to keep your mission-critical applications responsive.

[1] Electronic synchronization between the CPUs

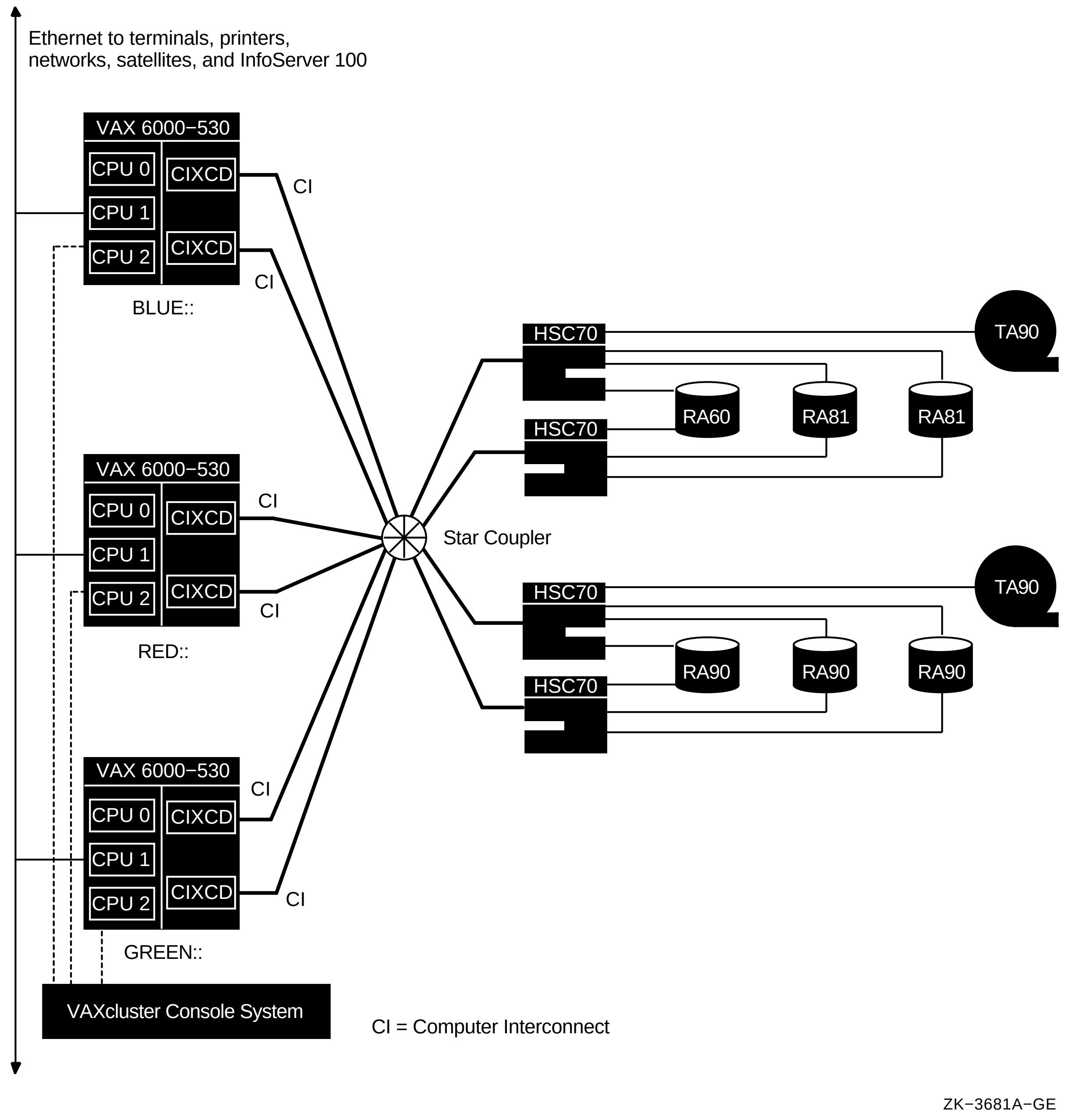
Rather than installing the exact number of system kernels required by the workload (n redundancy), you could install one extra system kernel of nominal capacity (n+1 redundancy). Then, when a kernel failed, your computing resource would still have sufficient capacity to provide adequate performance and responsiveness. Note that this approach works best when the system kernels in your loosely coupled system have similar capacities. If one kernel had as much capacity as the remaining ones, losing that single kernel could reduce your computing system's capacity by 50 percent. See Chapter 6 for more information on VMScluster systems, which use the loosely coupled approach.

4.1.1.4 Independently Recoverable System Kernels

A fundamental assumption behind having multiple system kernels is that they will not all fail simultaneously. Therefore service can continue when one of the system kernels fails. A related dependability feature of many, but not all, VMScluster systems and networks is the ability to recover or repair system kernels independently. This allows the VMScluster or network to be returned to full capacity without a total system shutdown. See Chapter 6 for more information on which configurations support independent system kernel recovery.

Figure 4–1 illustrates a configuration with independently recoverable system kernels.

Figure 4–1 A System with Independently Recoverable Kernels


Ethernet to terminals, printers, networks, satellites, and InfoServer 100
VAX 6000−530
CPU 0
CPU 1
CPU 2
CIXCD
CIXCD
CI
BLUE::
RED::
GREEN::
Star Coupler
HSC70
TA90
RA60
RA81
RA90
VAXcluster Console System
CI = Computer Interconnect
ZK−3681A−GE

The sample configuration in Figure 4–1 is a CI-based VAXcluster system. Note that each system kernel in the configuration has three identical CPUs. Such may not generally be the case, but the sample configuration simplifies the capacity calculations without obscuring the concepts.

Another simplification in this discussion is to focus solely on the CPU capacity of the system. In a well-tuned and balanced system, there is sufficient I/O bandwidth and memory capacity to support the workload. Keeping those factors constant, we shall examine what happens to the overall system's processing capacity when a CPU or a system kernel fails.

Figure 4–2 shows the impact of individual CPU outages and kernel outages.

Figure 4–2 Individual Kernel Capacity Versus Time: Example State Diagrams

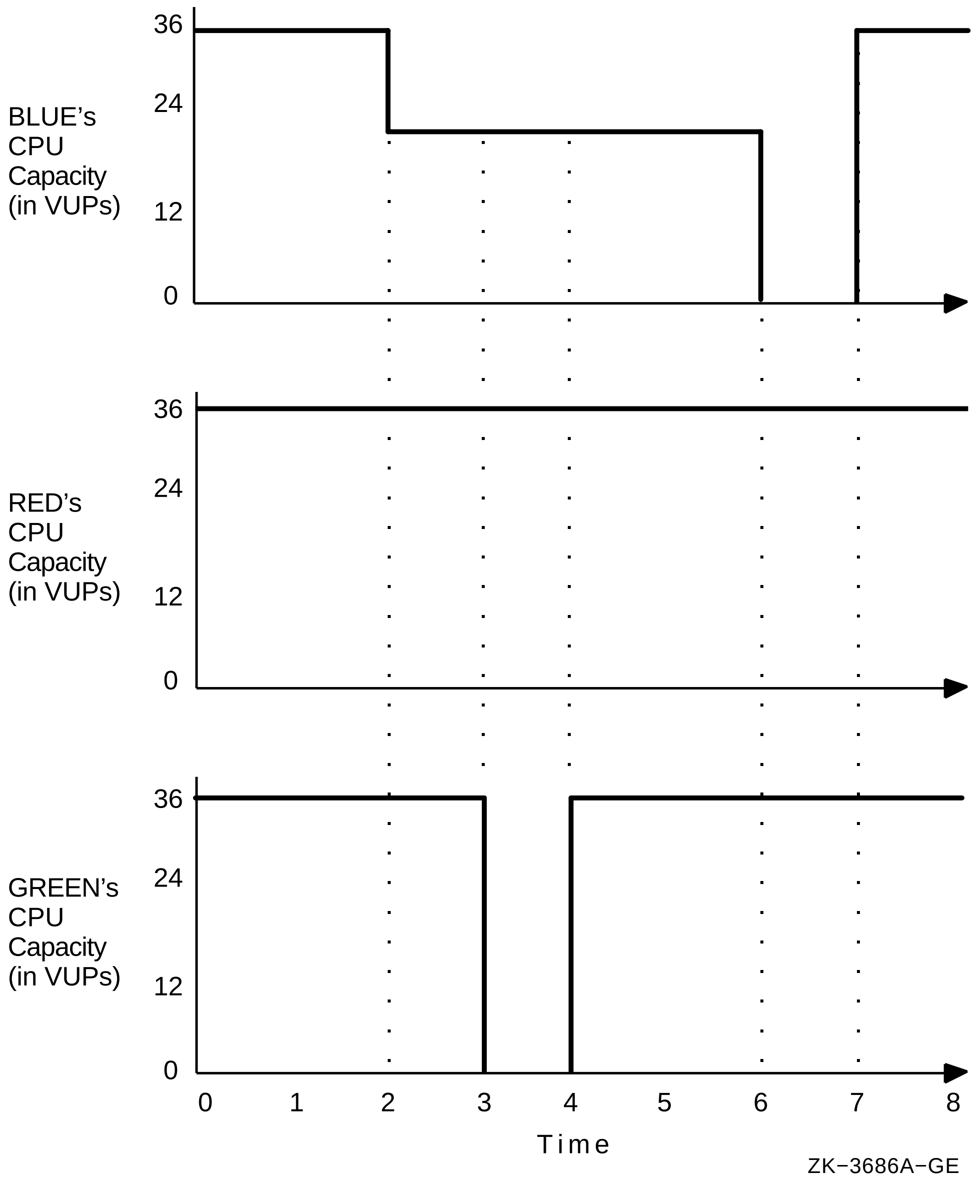


Figure 4–2 has three parts, and each part gives the CPU capacity of its system kernel as a function of time. In other words, the top graph in Figure 4–2 represents the CPU capacity of the top system kernel in Figure 4–1. The middle graph represents the middle system kernel and the bottom graph represents the bottom system kernel's CPU capacity. The CPU capacities are expressed in VUPs.

A **VUP** is a VAX unit of processing. For reference, one VUP is equivalent to the processing power of a VAX–11/780™ . Clearly, modern VAX CPUs (and the AXP processors) are much more powerful than the original VAX–11/780.

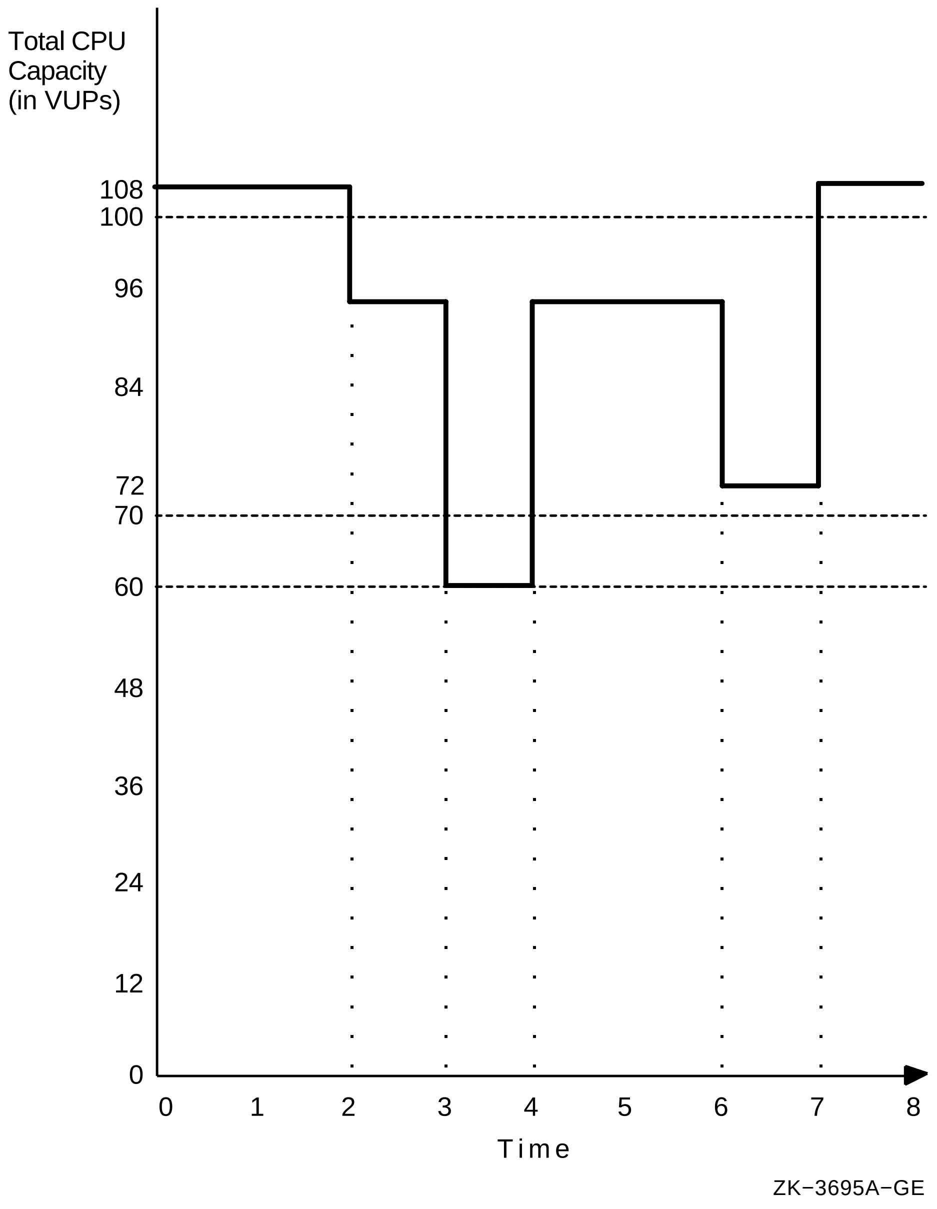
For simplicity, each system kernel in Figure 4–2 is identical. All three VAXcluster nodes are 3-processor SMP VAX 6000 systems; specifically, model 530. The total capacity of all nine CPUs in the system is about 108 times that of a VAX–11/780.

Following are the significant events along the time line in Figure 4–2:

- For time 0 < t < 2, full CPU capacity is available for your user's applications.
- At time t = 2, one of the CPUs in node BLUE fails, but OpenVMS reconfigures itself to proceed without stopping, using the remaining two CPUs. The process that is executing on the failing CPU is deleted.
- At time t = 3, node GREEN is halted as part of a major software upgrade.
- At time t = 4, node GREEN is rebooted to finish the software upgrade.
- At time t = 6, node BLUE is powered off to allow CPU replacement.
- At time t = 7, node BLUE is rebooted after the failed CPU has been replaced.
- For time t > 7, the overall system is operating at full CPU capacity again.

Figure 4–3 shows only one graph, representing the total available CPU capacity of all three systems at any point in time.

Figure 4–3 Total System Capacity Versus Time: Example State Diagram



In Figure 4–3, the total is calculated by finding the sum of all the CPUs that are on line for each time period, as time progresses from 0 to 8.

If your mission-critical applications require only 60 VUPs of CPU capacity to execute effectively, then the example system has had sufficient CPU capacity continuously available the entire time. That is, the solid graph line never dips below the horizontal dashed line at VUPs=60.

If those applications require 70 VUPs of CPU capacity, then time $3 < t < 4$ exhibits poor performance because only 60 VUPs are available. Clearly, BLUE's failed CPU should be fixed as soon as possible. Or, perhaps the software upgrade of node GREEN should be postponed until both nodes BLUE and RED are operating at full capacity.

If it takes 100 VUPs of CPU capacity to service your mission-critical applications effectively, then only time t < 2 and time t > 7 have acceptable performance characteristics. Either more CPUs could be added to the system kernels, or perhaps a similarly powered VAX node could be added to the VAXcluster. Also, AXP nodes could be added to this environment, making it a VMScluster with VAX and AXP nodes.

If it is critical that your system maintain full CPU capacity at all times, fault tolerant systems like VAXft systems should be used, perhaps in network configuration.

Note

The previous analysis has considered only CPU capacity requirements. Your applications also need memory and I/O bandwidth. Be sure to consider those areas as well when calculating how system capacity changes after a failure.

4.1.1.5 Network Level Redundancy

On a potentially larger scale, you can connect multiple independent systems (nodes) into a network that has pervasive capabilities. In this way, if one system fails, your users could access the computing resources of other nodes on the network. Adequate performance is an important characteristic of a dependable computing resource. Whoever configures your network should also consider the loading characteristics of the network when one or more nodes or interconnect paths has failed.

- Failure of an interconnect path

  Any properly configured wide area network (WAN) should have more than one path between any two network nodes. However, unless care is taken during the network design, the transmission and forwarding delays for a secondary path may be many orders of magnitude greater than for the primary path. This could be from routing the data through many additional nodes or from having to send the data across slower paths or both.

  The point is that even an adaptive routing network may require special *failed path* design considerations to provide continuously adequate service to your system users. See Chapter 10 for more information on Digital services that can provide expert help in configuring your network.

- Failure of a network node

  Many times, the failure of a key node can have a greater impact on a network than the failure of any of the paths. If the node is a router for many paths, the topology of the network is significantly altered. At least as important are the lost capabilities that the node had provided to the network system. This loss places a greater load on other network nodes, which must provide those services to your system users, assuming they also have those capabilities. There are many approaches to fully distributing your computing resources; all of them require that you *plan ahead*.

  For details about Network Application Support (NAS) from Digital, which provides pervasive capabilities for many vendors' systems, see the following documents:

  – *NAS Guide to Designing a Portable Programming Interface*

  – *NAS Guide to Documenting Multiplatform Products*

– *NAS Guide to Developing Portable Software*
– *NAS Guide to Using a Portable Programming Interface*
– *NAS Overview*
– *NAS Application Style Guide*

See the *DECtp Desktop for ACMS Training Guide* and the *DECtp Desktop for ACMS Progamming Guide* for more information on DECtp™ Desktop for ACMS. Also called Desktop ACMS, this software product provides enhanced transaction processing support for personal computer users.

See *Using CDD/Repository on VMS Systems* for information on CDD/Repository, which provides fully distributed, coordinated data repository functions.

4.1.1.6 Operating System Performance

Generally, a computing resource that is set up for maximum dependability also has a well-tuned operating system. This characteristic is typically the case because the operations involved in tuning an operating system (such as eliminating bottlenecks) also *usually* enhance the system's dependability. Enhancing the system's security (to avoid denial of service or data corruption) is one exception. Higher security generally involves more overhead, which can degrade performance in some cases.

Sometimes failures can cause the system operations to move away from the most optimal settings. The OpenVMS operating system has several mechanisms for providing continuous service in the presence of some failures, but at degraded performance. Examples include:

- An SMP configuration continuing after a CPU has failed (a significant impact unless an extra CPU was configured: n+1 redundancy)
- A CPU continuing with its high speed cache disabled (a very significant impact, for that CPU)
- A system kernel continuing with some of its main memory disabled (the impact depends on the resulting level of memory use)
- A single HSC™ or HSJ controller continuing to handle the full I/O load after its twin has failed (the impact depends on the resulting I/O load on the HSC or HSJ and its disks)
- A VMScluster continuing with one path of its dual-rail[1] computer interconnect (CI™) disabled (usually not a measurable impact on performance)

All multiuser operating systems protect themselves from nonprivileged users and programs (to a greater or lesser extent) and these dependability features require a small amount of system overhead to operate. Some operating systems have internal firewalls, which can protect the inner executive from the failures of even other privileged subsystems. This is useful when extensive systems programming is required of a customer's staff for effective system use. There is, however, a substantial overhead involved with firewalls, and they can make interapplication communication a very complex undertaking. Systems programming is rarely required to fully use OpenVMS so the overhead of firewalls has been avoided.

[1] The CI uses 2 send cables and 2 receive cables per connection.

4.1.1.7 Application Software Performance

Some high availability features of Digital application software products (such as online database redefinition or online application maintenance) generally have minimal impact on the product's performance. Other features, such as journaling or online backup functions, can impose an additional I/O load on the storage subsystems.

Recovery operations (such as the automatic rollback of transactions after a system failure) can temporarily impact the performance of production work, but such overhead is usually not encountered during normal operations.

One effective way to ensure that your system's users can always access applications is to use one of the Digital transaction processing (TP) environments. Because DECtp products enable the implementation of distributed TP applications, your applications can benefit from VMScluster system or network levels of redundancy, or both.

Because application performance is so critical to TP environments, the DECtp products make optimal use of OpenVMS features. The result is that most customers find that writing their applications for the TP environment generally increases the level of performance over a non-TP environment.

For information about designing and implementing a transaction processing (TP) system, with ACMS, data dictionary, DEC Rdb, DECforms examples, see *A TP System Case Study: Database and Application Development*. For information about the Digital relational database product, see the DEC Rdb documentation set, including the:

- *VAX Rdb/VMS Guide to Using RDO, RDBPRE, and RDML*
- *VAX Rdb/VMS Guide to Database Design and Definition*
- *VAX Rdb/VMS Guide to Database Maintenance*
- *VAX Rdb/VMS Guide to Database Performance and Tuning*

Also see the *VAX ACMS Guide to Creating Transaction Processing Applications* for details.

4.1.1.8 Personnel Productivity and System Performance

A good way to reduce the number of mistakes your personnel may make is to reduce how much they have to do. (See Chapter 11 for a related case study.) This improvement can be done with more automated procedures and can increase, not only the reliability of your personnel, but their productivity as well.

The drawback of some software procedures is that they can increase system overhead as they perform automatically all those tasks that your personnel used to perform manually.

4.1.2 Implementation Tradeoffs

This section covers the feasibility, timing, and learning curve considerations of implementing improvements to your computing services.

If ever there were a time to *plan ahead*, this is it.

4.1.2.1 Feasibility Considerations

Read again your listing of the constraints under which you must operate. Perhaps conditions have changed and some constraints may be a bit more flexible than the last time you checked. Of course, new ones may have developed as well.

If the set of dependability requirements your company places on your computing resource cannot be properly addressed because of organizational or cultural factors, it may be time for some creative management education.

If insurmountable constraints prevent you from making those changes in your computing resource that are necessary to provide the required level of dependability, you should at least make the consequences clear to the appropriate managers and be sure that the expectations of your users are set correctly.

4.1.2.2 Timing Considerations

Funding constraints are likely to limit the rate at which you can enhance the dependability of your computing resource.

In making choices about how to enhance your system, you will have to get to know your users. Are they likely to complain about a system clearly in need of more capacity but always available to them? Or will they be able to live with a system that has good response times, except when it is unavailable?

In most situations, you cannot concentrate on only one system attribute to the detriment of others. You may have to select the most pressing performance improvement, with the most critical dependability enhancement, along with the most needed programming tool, and so on.

And then next year, review and revise all your lists again when you submit your long range plan to management for funding.

4.1.2.3 Learning Curve Considerations

The effort to improve your computing resource in all dimensions takes time. It takes time not only to implement the actual technical changes, but for the operational procedures to evolve accordingly. Your computing system responds to changes in your company's business practices. At the same time, your company's practices are perturbed any time you make significant changes in your system.

Besides considering your personnel's training requirements, you should remember that your system's users also have a learning curve to climb. The success of your new system depends on your personnel's ability to use it effectively.

Finally, once your company and system users have become accustomed to their more dependable computing resource, present them with *before* and *after* information that demonstrates the return on their dependability investments. This step might make it easier for you to obtain funding for improvements in the next year.

4.1.3 Staffing Tradeoffs and Considerations

As you enhance the dependability of your computing resource, your staffing needs will keep changing as well. Staffing considerations are intermingled with other considerations because it is your staff that ultimately determines how successfully your computing resource provides service to your users. The following list contains examples of staffing considerations:

- You may opt for the simplicity of acquiring a fully fault tolerant system, rather than hiring a consultant to craft an active-standby system from existing components. See Chapter 6 for more information on Digital VAXft fault tolerant systems.

- You may decide to hire (or train) TP programmers to implement an online solution, rather than hire more operators and maintenance programmers to keep the current batch-oriented solution running. See Chapter 8 for more information on transaction processing environments. See Chapter 9 for more information on the DECscheduler job control system.
- You may determine that hiring a database administrator and acquiring a PC integration package would support your users' thirst for data more effectively than the ad hoc remedies maintained by your system managers and programmers.
- You may discover that giving your best operator *system management* training provides you with the perfect *chief system monitor* for the control room of your new *lights out* data center. See Chapter 11 for a case history of how the Digital Customer Support Center in Colorado implemented its *lights out* data center.

4.1.4 Vendor Tradeoffs

Because high quality service can mean the difference between a minor outage and a major disaster to your system users, effectively managing your company's relationships with the vendors who service your hardware and software is critically important.

4.1.4.1 Reliable Products

Clearly, the quickest service call is the one that never happened. Just as clearly, even the best service vendor has trouble providing satisfactory results if the products it is servicing require frequent or major corrective actions or both.

4.1.4.2 Adequate Coverage

If your computing system must be available to your users 168 hours per week, then your service vendor should be available to help you that number of hours as well.

An important question is whether 24-hour support implies a telephone answering service or a senior technical support person at your system site at any time.

4.1.4.3 Committed Response

Committed response should mean more than a maximum time for a callback. Committed response should also imply an operational procedure to escalate the repair effort on a predefined schedule if the complexity of the problem requires it.

Having the service person desert your (still broken) system at 5:00 p.m. is not the best time to find out that your service contract does not include a continuous effort clause. A **continuous effort** clause means that once a service call has begun, the vendor will not cease working on the problem until it is fixed.

4.1.4.4 Location Flexibility

You should find out if your service vendor can place a field engineer in residence at your site to respond to your most mission-critical problems. Likewise, find out if your service vendor can accommodate your company's remote sites, which may be scattered around the country (or globe).

Of course, just as important as the location of service staff is the location of spare parts that may be needed.

4.1.4.5 Flexible Roles

One way you can save service costs is to elect to have your own staff service the system, if you have the expertise in-house. Determine if your service vendor can provide appropriate training, spare parts, and backup support to help you succeed in this type of effort.

On the other end of that spectrum, does your service vendor have the resources and experience to provide out-sourcing services to your company? In this scenario, the out-sourcing vendor accepts complete responsibility for the effective operation of your company's computing resource.

4.1.4.6 One-Stop Service

Because all the components of your system must function together to provide service to your users, it makes sense to service them as a team. If your service vendor has the capability to bring the many pieces of equipment in your system into an umbrella service agreement, it eliminates the *finger pointing* that can develop. For instance, sometimes two boxes on a network cannot communicate effectively, but still appear to be functioning separately. Is it box A's problem? Or is it box B's problem? Or is it a problem somewhere else in the network?

See Chapter 10 for more information on Digital services.

4.1.5 Cost Tradeoffs

Many times, the best way to enhance the recovery capabilities of a system is by adding redundant equipment or replacing some parts of it with fault tolerant equipment. Other times, acquiring the appropriate software products is an effective route to a more robust system. Of course, having the right type and number of people on your staff is key to providing dependable computing but this also costs money.

In the final analysis, the essential bottom line consideration is the fiscal one. It would not be rational for your company to invest in dependability enhancements unless they were well justified. The following steps may help you justify an appropriate level of funding:

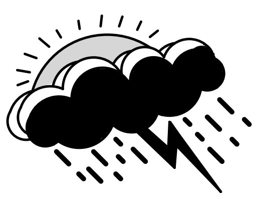
1. Use the information you recorded in Table 4–1 and make a list of the most important mission-critical business functions.
2. Find out, from the relevant users, just how expensive an outage is for them. Include rework overhead, lost revenue, and the loaded labor costs of any workers idled during an interruption.
3. Analyze your computing system (with the help of this handbook) to determine what, if any, improvements need to be made so that its dependability characteristics match the dependability needs of your company.
4. Compare the improvement costs over five years to the outage costs over five years. You can use the past history of your current system to make an estimate of the frequency and duration of production outages, given the current system.

   Few companies have such high outage costs that a *blank check* for dependability enhancements can be justified. However, you can use the previous information whenever you are requesting funds for other purposes, such as increased capacity. There may be only a small difference in total costs between enhancing your system in a conventional way and doing so in way that enhances its dependability.

5. Consider that the best way to improve your system is by defining and securing funding for a project dedicated to improving its dependability characteristics.
6. Keep careful records of outages, costs, and the installation of dependability improvements. By documenting your successes and mistakes, you can track your system's progress and more easily demonstrate to management the return on their investment due to fewer and shorter outages.

5

Maintaining a Dependable Environment



The physical environment should not be taken for granted when planning computer systems. It is tempting to assume that if the already existing computer room has the space, the electrical power and air conditioning can handle the new load. Even if such details are calculated and found to be adequate, do not assume that such critical resources would never fail or degrade.

Newer computing equipment generally requires less power, generates less heat, and occupies less space than older equipment, assuming equivalent computing capacity. However, the “older” equipment may be only a few years old and may not be sufficiently capitalized to replace. Then the old and the new must share site resources that may have been initially planned to support only the old configuration.

Because your computing system is built completely on top of its environment, it can never be more dependable than its environment. If your business requires a level of system dependability greater than that provided by your utility companies, you must take steps to augment the dependability of your system’s physical environment.

This chapter discusses many key environmental factors with the goal of facilitating your analysis process. The factors discussed are:

- Electrical Power (Section 5.1)
- Air Conditioning (Section 5.2)
- Water Supplies (Section 5.3)
- Site Security (Section 5.4)
- Desktop Environments (Section 5.5)
- Dealing with Personnel (Section 5.6)
- Coping with Disasters (Section 5.7)
- Overall System Considerations (Section 5.8)

5.1 Electrical Power

When the power supply fails, chances are good that your computing system will fail unless you have taken extra precautions. Your main power supply must not only be resistant to power outages, but it must also be free from “noise” which causes an unwanted disturbance of data due to power fluctuations. Voltage spikes, brown outs, and frequently missed voltage cycles can put undue electrical and thermal stress on the circuitry in your equipment. This stress can cause premature failure of many solid state components.

To combat problems with power fluctuations, many vendors offer power conditioning and distribution units, such as the following Digital products:

- H7007 TVSS (transient voltage surge suppresser)
- H7225/H7226 CVC (constant voltage conditioners)
- H7228 PCS (power conditioning and distribution)
- H7317 PDS+ (power distribution and monitoring)
- H7318 PCS+ (power conditioning, distribution and monitoring)
- HA3000, H7229 (uninterruptible power systems)
- H7310 EMS (environmental monitoring system)
- VAX REMS (remote environmental monitoring software)

Some high-end systems, such as the DEC 10000 AXP and the VAX 10000, have built-in power conditioning units (like the PCS+).

The combination of the various H731x-family units and the OpenVMS layered product VAX REMS provides a comprehensive environmental monitoring system. From a single terminal, your staff could monitor over 100 sensors that would report on and record the following environmental conditions:

- Voltage
- Current
- Temperature
- Humidity
- Smoke
- Fire alarms
- Water detectors
- Air conditioners
- Door alarms
- Security systems
- Backup diesel-fuel reserves
- Backup power systems
- Halon systems
- External electrical panels

For each environmental condition in the previous list, various automatic system actions such as sending electronic mail could be triggered on alarm conditions.

There are also a number of ways to protect yourself from total power outages of your utility power. For short power outages of a few minutes to several hours, you can purchase one of the many types of uninterruptible power systems (UPS). These units are capable of providing power for systems ranging anywhere from a single desktop computer to several large systems in a computer room. Some systems, including the Digital VAXft series of fault tolerant computers, use built-in UPS units to avoid down time from power failure. Digital also offers separate UPS products for computer rooms and offices.

Note

It is possible for UPS units to fail precisely when they were needed. One rather expensive way to avoid such a disaster is to install redundant UPS units. Another effective and less expensive way to test the proper functioning of your UPS unit is to actually bring the UPS unit on line by performing practice drills. If you have practice drills, remember to refill the engine's fuel tank after each online drill.

To survive longer power outages of days or weeks, you might invest in a gas or diesel powered engine-generator unit that could recharge the batteries of your UPS unit. Many hospitals install these types of emergency generating facilities.

Installing an engine-generator unit is very effective for most applications but for some applications this might not provide enough protection. For mission-critical applications, you might consider one of these options:

- Installing a second computing site that receives power from a different power grid.

  Even though installing a second computing site is the more expensive option, installing the second site in a geographically distant location can also provide you with disaster tolerant capabilities (see Section 5.7). Building and staffing a second site may not be required if the only reason for adding the remote site is access to a second power grid. A UPS unit is a more cost effective solution to that problem.

- Contracting with your power company to route a feed from a second grid into your current computer site.

  When one power grid fails, there is a good chance that a second, independent power grid can still provide power to the computing system. This option can be less expensive than building a second site.

5.2 Air Conditioning

High temperatures are the enemy of solid state circuitry. When solid state devices operate at elevated temperatures, they are more likely to fail. Therefore, it makes sense to acquire extra air (or water) conditioners. For the purposes of surviving failures, it is better to achieve a total cooling capacity in several smaller units than in one or two very large units. Then, if one unit fails, a smaller percentage of the total cooling capacity is lost.

If you have installed multiple air conditioning units and the alarm on your monitoring equipment signals a problem with an air conditioning unit, consider following these procedures:

- If the air conditioners that remain working have sufficient capacity to cool the total system, you can continue to offer full computing services while waiting for the service vendor to repair the broken unit. You can also ensure proper distribution of cool air throughout the computer room by using portable fans to compensate for disabled blowers.

- You can partially shut down the computing system to reduce the amount of heat generated and thus reduce the total amount of cooling required. Of course, shutting down computers and peripherals is possible only when the applications affected are not mission-critical. As you shut down equipment and the total capacity of your computing system is diminished, you may also need to shut down noncritical applications in order to achieve reasonable performance on the critical applications.

Clearly, the order in which you shut down equipment and applications can become controversial. Therefore, you should negotiate the shutdown procedures before cooling problems occur. Planning ahead for an orderly shutdown reduces the probability of losing work, data, or equipment.

Note

If your air and water conditioning equipment is operating with electricity (and most are), it makes sense to provide UPSs for them as well. Providing a UPS for the computing equipment and letting the cooling equipment stop when the building power fails could require shutting down the system anyway and could potentially waste the investment of the UPS (unless all power failures were guaranteed to be extremely short).

5.3 Water Supplies

Even if your computing equipment is air cooled and not water cooled, you may still need to worry about plumbing. This is because many air conditioning units transfer heat from their computer room blower units to their externally located radiating units via pumped water.

Environmental monitoring systems typically have water sensors that can be located in out-of-sight places (such as under raised computer room floors) to alert you if water or other liquids have begun to leak there. These are good investments because electrical wiring has a habit of shorting out when it gets wet and damaging the equipment to which it is attached.

Note

A true story: one installation believed “the colder the better” should apply to computer rooms. Therefore, it decided not to replace its very powerful air conditioner when its high heat generating CPU was replaced with a cooler system. Because people rarely entered the room, it was assumed that there was no disadvantage to having the new computer in an “ice box.” When all the water pipes in the room froze and burst, causing considerable damage, the installation decided to stay within its vendor’s recommended ranges of temperature and humidity.

5.4 Site Security

Some of the security mechanisms provided by computer operating systems rely on adequate physical security for key hardware components, such as the Power and Reboot switch on the CPU cabinet. To deny service to unauthorized persons, you must control access to those pieces of equipment. Many companies offer products, such as badge reader door locks, that provide controlled computer room access in a flexible fashion.

Note that people and objects are not the only external influences that can intrude on your system from the external environment. Destructive programs (known as viruses because of their ability to spread around a network) can be dangerous to the software and data in your system. In addition, malfunctioning external equipment can impact your computing system just as seriously as a local power failure. See Chapter 9 for more information on system security.

It is possible, however, to carry physical site security too far.

Note

A true story: a fire detector was accidentally tripped in a highly secure computer room. Because the room was actually an electronically controlled vault, the operators were required to perform a 15-step checklist to allow them to escape before the Halon fire control system forced all the oxygen from the room.

In other words, remember that personal safety is more important than computing system safety.

5.5 Desktop Environments

Be sure to consider computing equipment in office areas. You may succeed at optimizing the conditions in your computer room; however, if your VMScluster system has satellite nodes that reside on or beneath people's desks, you must consider temperature, power, CPU console security, and coffee spills in the office environment as well. For example, perhaps you are already using some type of Halon system in your computer room, but have you overlooked the option of providing a fire extinguisher in every office with a desktop or personal computer? You should also consider the consequences to your office equipment in case a fire turns on the overhead sprinkler system.

Note that similar considerations apply whenever significant parts of your computing system such as CPUs are in employee work areas. Physical environments such as shop floors, checkout lanes, and loading docks all contain special and unique environmental hazards. See Chapter 10 for information on how Digital can help you address those hazards.

In addition, even if you transport special sets of full backup tapes from your data center to a fire storage facility for safekeeping, are you sure that users are backing up and safely storing corporate data that may reside only in their personal computers hard disks or diskettes? See Appendix A for a checklist that can help you identify similar areas of concern in the office environment.

5.6 Dealing with Personnel

As hardware and software reliability has improved over the years, the percentage of down time due to those factors has decreased and the impact of operator errors or system management mistakes has increased. Much of this increase is the result of a very rapid growth in the power and complexity of the “average” computing system.

Just a few years ago, it might have been reasonable to expect one competent systems person to manage, operate, and even sometimes program a Digital MicroVAX II™ system that a company might put in the corner of the office. If that MicroVAX II is replaced with a newer, cheaper, and smaller Digital VAX

4000 system or a DEC 3000 AXP system, that same systems person could be supporting the computing needs of over 100 users instead of a dozen.

As the labor costs of competent personnel keeps rising, you need to analyze how to make the most efficient use of your staff. The following sections outline some techniques you may wish to investigate. See Chapter 9 for more information on efficient operational procedures.

5.6.1 Comprehensive Training

While training cannot eliminate certain classes of human errors (such as typographical mistakes), it should make your staff aware of the impact of their actions (such as dismounting the quorum disk). In-depth training in how your computer system operates has additional advantages. If your operations personnel have sufficient knowledge about *how* their standard procedures function and *why* they were designed that way, those people are more likely to suggest useful improvements. Additionally, there may be crisis situations when the maintainers of the procedures cannot be reached and the attending staff has to implement temporary workarounds to finish the critical jobs on time.

Continuing education is a positive dependability factor for all your personnel. System managers and application system designers need to be aware of any new dependability features of your vendors' computing hardware or software to use them to maximum advantage. Applications programmers need to keep up with the latest techniques for defect-free programming and friendly user interfaces. Appropriate end-user training can reduce the load on your data center's help desk. See Chapter 10 for more information on training options available from Digital.

5.6.2 Suitable Tools

A wise person once said, "A craftsman never complains about having the wrong tool." This is because all the proper tools are assembled before even starting the job.

Your computing system is itself a highly complex tool. The right tools can help your staff maintain proper control of the system and its functions. Without system tuning tools, your staff may be installing more hardware than necessary to maintain the required responsiveness. Without data repository tools, your staff may be spending too much time finding and fixing subtle application defects that are caused by using slightly different data definitions in different programs or on different nodes. Without online documentation tools, your staff (and users) may be interacting with the system without understanding all of the options available to them.

See Chapter 8 and Chapter 9 for more information on tools that could enhance your staff's ability to program and operate your computing system in a dependable fashion.

5.6.3 Order and Neatness Contributing to Safety

When your computer equipment is installed in neat geometrical arrangements, it is more than pleasing to the eye. It is also more reliable and more recoverable. People are less likely to trip over cables or pipes when they are routed through cable trays or under a false floor. So it is safer for them and safer for your system.

When equipment is arranged according to the specifications supplied by your vendors, it is easier and quicker to fix. It is also generally the case that installation specifications also consider critical reliability areas, such as proper air or water flow and distance from electrical interference.

Note

Even tightly cramped computer rooms can have full-height (2.4m) ceilings. If some of your equipment is half-height (1.2m) or shorter, you may be able to save floor space by stacking cabinets with appropriate racks or shelves.

5.6.4 People-Proof Covers

When people move around or stand in close proximity to computing equipment, there is always a chance of bumping against a critical button or flipping the wrong switch accidentally. Many vendors build console lock-out systems or secure cabinets into their systems. There are also many “aftermarket” accessory vendors who supply user-installable covers and latches that prevent the accidental spinning down of the system disk or the mistaken pressing of the Reboot button.

The point of using covers and latches (even keyless ones) is to require the person to slow down and engage in a short multistep process before any critical action can be completed. Clearly, if the Exit button for the electric computer room door is on the same wall as the Fire Alarm or Power Shut Off buttons, the Exit button should not be covered but the others *must be.*

5.6.5 Operational Zones

If all the line printers and laser printers were set up in a different room than the data center, the print operators would seldom have reason to enter the room with all the critical switches and buttons. In fact, if you can restrict access to the core part of the computing system, there would be fewer chances for human error to impact it.

Implementing a zoned environment involves working out the roles of each person on your staff. Some negotiation may be necessary to establish clear boundaries of roles. After the role boundaries have been created, the decision of where to place the physical boundary becomes straightforward. Note that role boundaries do not preclude cross training. People might want to shift roles some day or might need to cover for each other during sick time.

If budget or other constraints prevent the construction of actual physical boundaries, many times rearranging the equipment in the data center can provide the necessary virtual boundaries. The point is for people to go near only those pieces of equipment for which they are authorized.

5.6.6 Lights Out Computing

The ultimate safe zone is a computer room with no people around. Then you could turn the lights out because the equipment would not need it. The challenge in lights out computing is to automate the system’s and business’ procedures to the extent that human operators are no longer needed.

A central control room can be used to enable a small number of professionals to monitor and control large numbers of computers. One Digital product, VMScluster Console System (VCS), is an effective way to connect “directly” to the console lines of a large number of systems. When VCS is used in concert with other Digital products such as DECalert™ and DECscheduler, your staff can automate a large number of routine operations tasks. If a problem develops, the responsible staff members or a predefined chain of backup persons would be the first to know.

Note

In February 1994, Digital granted Target Systems Corporation a nonexclusive, nontransferable license to use DECalert Version 1.2 software. Under the agreement, Target Systems may provide support for the Digital customer base (existing and future) for the product, as well as future development and migration of the product or its derivatives. Target Systems named the product TARGET–>ALERT.™ Digital reserves the right to continue to sell DECalert or its enhancements, as Digital deems fit.

See Chapter 9 for more information on the previously mentioned products. See Chapter 11 for a case history of one of the lights out data centers within Digital.

5.7 Coping with Disasters

The term *disaster* usually brings forth images of floods, fires, and similar destruction. Such natural occurrences can prove to be disastrous for a computing installation but they are not the only potential sources of catastrophe for your system.

There are many hazards that can pose a threat to your computing systems. Examples of hazards include a disgruntled employee who erases critical computer data, a virus program introduced into your network by a naïve PC user, or a construction worker severing the wrong cable. All these scenarios can put your system out of commission in completely unforeseen ways.

Because it is impossible to enumerate all the potential causes of disaster, an alternative strategy is to define broad classes of problems that could be addressed by several general classes of contingency plans.

Note

As illustrated in Section 5.7.3, effective disaster planning is neither inexpensive nor easily accomplished. This section is an overview of the issues involved and *is not* a step-by-step guide on how to accomplish disaster planning.

If your computing resource must be truly disaster tolerant, be sure to consult disaster/risk management professionals. See Chapter 10 for more information on Digital services.

5.7.1 Time Domain Considerations

A key dimension of coping with disasters is timing. The questions that must be addressed by your contingency plans include:

- When disaster strikes, what is the maximum allowable delay before service must be restored to key applications?

  Clearly, the longer you can safely wait to restore service, the more options you have. If three or four days down time is acceptable, it might be possible to make ad hoc arrangements with your vendors for replacement equipment or sites. If your business cannot lose more than a few minutes of availability, you must be prepared to sustain the additional costs of hot standby sites.

- When service is restored, how current must the files and databases be?

  In other words, how much reentering of data or transactions is acceptable? A requirement of not losing a single transaction places more severe constraints on the final disaster tolerant solution. Therefore the solution may have to be justified by the business costs of not meeting such a requirement.

- What level of performance degradation is acceptable when the backup system is carrying the load?

  One way to reduce the cost of maintaining a hot standby site is to limit its capacity. The impact of that strategy may be longer response times to interactive users, a policy of executing only critical applications, or both.

- How long would it take to assemble the staff required to bring the backup system on line?

  This issue involves a staffing decision: to maintain a full-time staff at the backup site, to issue beepers, to telephone people at home, or to wait until the next business day to reassign tasks. Clearly, the acceptable delay is determined by your unique business context.

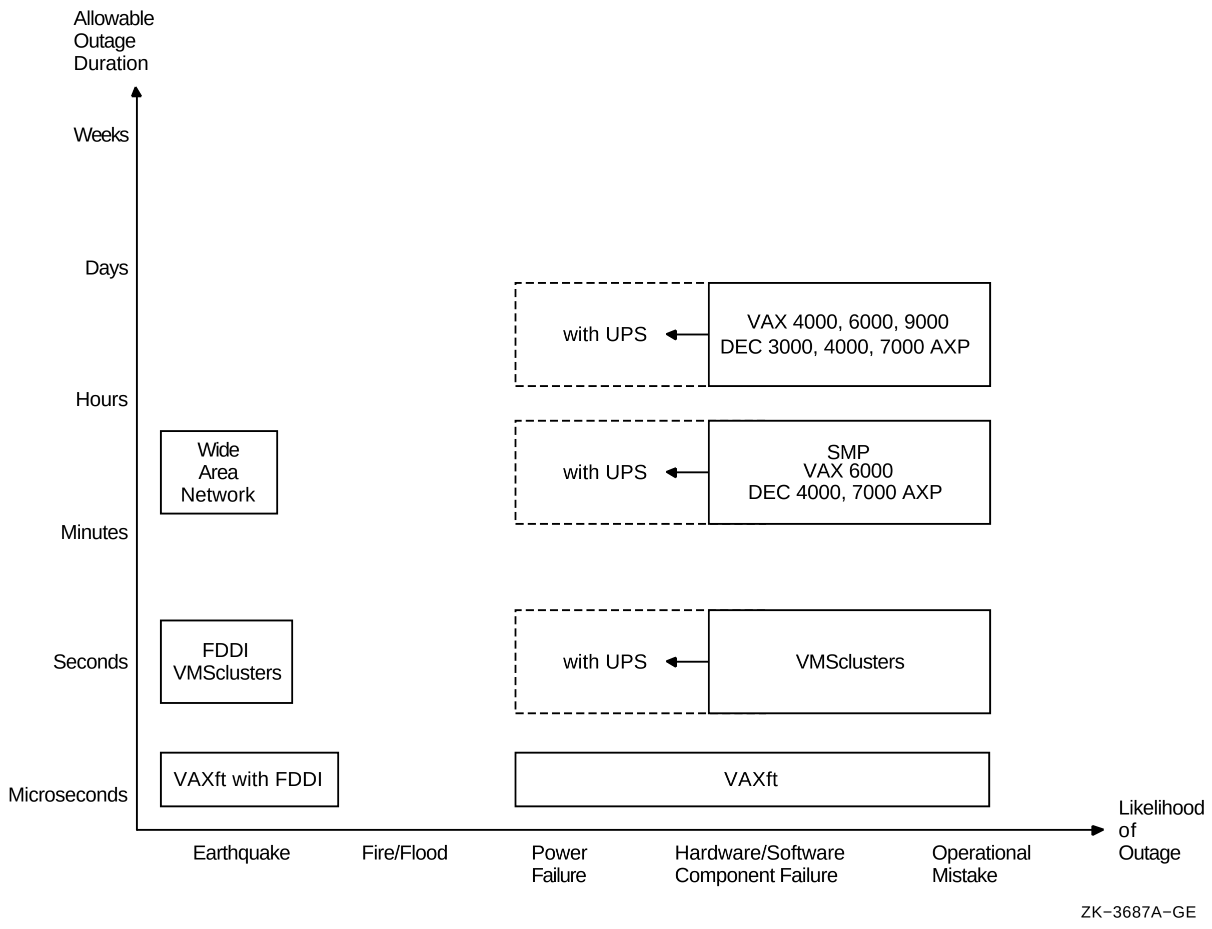
- How quickly can vendors deliver replacement equipment if required?

  Some classes of disaster could be handled with prompt delivery of new hardware and software. This assumes the physical site was not completely destroyed or could be rebuilt quickly. Again, *quickly* can only be defined in your business context. This also assumes that your vendor can provide identical hardware and software revisions and that your applications and data can be retrieved from secure storage.

- Do your disaster recovery procedures require an efficiency of execution attainable by most human operators? What are the required time constraints on your staff? Must they perform their tasks at super human rates or levels of perfection to enable your contingency plans to succeed?

Figure 5–1 illustrates how various AXP and VAX system configurations respond (in the time domain) to outages. See Chapter 6 for more information about the behavior of various AXP and VAX configurations.

Figure 5–1 Time Domain Behavior of AXP and VAX System Configurations



In Figure 5–1, the horizontal axis lists various causes of system outage, mostly of an external environment nature. It is assumed that the events listed near the vertex of the two axes are extremely rare while those toward the right of the vertex are progressively more commonplace. The boxes in the center represent typical AXP and VAX systems that could not survive a power failure *unless* they were connected to uninterruptible power systems (UPS). The system configurations that are spread over a wide geographical area such as VMScluster environments using FDDI or wide area networks (WANs) are more likely to survive rare but catastrophic disasters.

The vertical axis in Figure 5–1 indicates how long typical recovery times are likely to take. The symmetrical multiprocessing (SMP) AXP and VAX systems are shown to have shorter recovery times than the non-SMP configurations because a system with a single CPU has a critical single point of failure.

An SMP AXP system or SMP VAX system could continue in degraded mode on its remaining CPUs (perhaps after a reboot), while a non-SMP AXP system or a non-SMP VAX system would be inoperative until the service person could replace the failed components.

The VAXft systems have recovery times in microseconds because of their hardware-based fault-tolerant architecture. This capability makes them very useful in real-time or transaction processing applications.

5.7.2 Hot Standby Sites

If your computing system supports a business function that is so critical that it must be disaster tolerant, consider using a hot standby site that can take over production operations if your primary physical environment becomes unusable. Typically a hot standby site is in a physically remote site.

The following list describes several options for providing emergency computing resources for your company:

- Mutual backup processing agreements

  Some companies enter into mutual backup processing agreements with other companies that use similar hardware and software configurations. In the event of an emergency, the site that is still operational provides computing resources to the other company to execute critical applications. This mutual arrangement works well between different companies as well as between different divisions of large corporations that support division-level computing centers.

- Duplicate computing facilities

  If performance or security requirements are a high priority for your company, it might not be possible for you to share another company's computing system. Instead, your company might have to incur the extra expense of installing a redundant computing system and providing it with a data center staff. The purpose of this duplicate computing facility is to be immediately available to take over the processing load in the event of a disaster.

  The difficulty of providing a duplicate computing facility cannot be addressed by this handbook because very subtle issues can make or break the success of such an endeavor. For instance, even if each physical site gets its power from separate power grids, the system's dependability cannot be guaranteed if the system users connect to the computing system using phone lines. In this scenario, a fire in the phone company's central office could be the hidden single point of failure that makes the entire system unavailable.

- Vendor-supplied backup sites

  Many computer vendors offer you the ability to use their physical site as a backup site on an as-needed basis. Using a vendor site can be less expensive than building your own backup site but your company gives up a certain amount of control to the backup site vendor. See Chapter 10 for more information about Digital services.

- Multiple-site data center VMScluster systems

  By utilizing a long distance FDDI technology, a multiple site data center VMScluster system lets you physically separate some of your VMScluster nodes up to a distance of 40 kilometers. This provides a measure of disaster tolerance. In these VMScluster systems, all nodes can produce useful work continuously until such time when a disaster at one of the sites removes one or more nodes. At that time, the surviving nodes may continue to provide service for system users without losing data or transactions. With any VMScluster system, you can also address performance requirements by:

  - Configuring extra hardware capacity into the system
  - Shutting down applications that are not critical to the success of the business

  - Tolerating degraded application response times until all sites become operational again

  See Section 6.4.4 and Section 7.4 for summaries of FDDI VMSclusters and the Business Recovery Server™ software product. Refer to *Managing the Business Recovery Server* and *Configuring the Business Recovery Server* for details about configuring, setting up, and operating FDDI VMScluster systems using the Business Recovery Server software product.

  See *VMScluster Systems for OpenVMS* and *Guidelines for VAXcluster System Configurations* for related VMScluster configuration information.

5.7.3 Business Considerations

Once all the technical details in your disaster plan have been addressed, the relatively easy part has been accomplished. The myriad other logistical details, which are usually business oriented, will be the main stumbling blocks to your computing system surviving a disaster. They are also the most difficult to enumerate in advance. The following questions include some business considerations:

- Does your company's insurance cover the disaster? Even if your plans fail?
- How would operating in the recovery mode affect your company's finances? Is it a high-cost scenario?
- Are there any clauses in your clients' contracts that would be violated during emergency operations?
- How much time and expense for training and drills is allocated in your operating plans?
- What provisions are made in your budget for the overtime that your staff would put in during recovery operations?
- How costly will personnel mileage reimbursements be if the entire data center staff must commute to a remote site for an extended period of time?
- Do your contingency plans cover the possibility of *vendor disasters,* such as disruptions in your service vendors' offices, central telephone office failures, or city sewer outages that can close the entire block?
- Could your company provide some level of disaster tolerance for your clients, such as spare personal computers or terminals to provide access to your system, even though your client's site was not usable?

How you address concerns such as these can mean the difference between a successful execution of an up-to-date disaster plan or a "false sense of security" that evaporates when an emergency strikes.

5.8 Overall System Considerations

You might find it useful not only to look at the effect of the environment on the computing system but also to look at the overall implications of environmental factors on the entire business. For example, suppose the primary mission of your computing system is to run an application that automates the manufacturing functions in the company's main factory. Investing in a UPS for your computer room would have little value if a power outage would shut down the factory equipment and the computing system would remain operational only to wait idly for work to do.

On the other hand, suppose that you have not invested in a UPS and after the utility power resumes service, the computer's manufacturing application takes significantly longer to restart than it takes the factory itself to restart. In this case, what does it cost the company if the factory has to remain idle while it waits for the computing system to restart the manufacturing application? From this point of view, it might be worthwhile to invest in a more robust environment. With a dependable system in place, your company's profitable production could be resumed sooner because your computing system would remain operational through the power outage and would be ready to resume processing the manufacturing application on demand.

6 Dependable Hardware Configurations

As discussed in Chapter 2, the dependability of a computing resource (or system) is affected not only by the hardware but also by the software, the physical environment, and the people who interact with it. It is usually the case, however, that the software configuration is determined to a large degree by the business functions to be performed by your computing resource. Also, the way your company needs to go about its business has a large impact on how you can respond to dependability requirements with staffing alternatives. The appropriate physical environment is largely determined by the needs of your system's hardware configuration.

These factors make the hardware configuration an independent variable that has a dramatic effect on the dependability of your computing resource. This characteristic is particularly true in the case of the AXP and VAX series of computers running the OpenVMS operating system, where the software environment can be identical regardless of the hardware configuration.

To take advantage of the enhanced dependability of a different software technology, such as a transaction processing (TP) environment, you may need to rewrite your application programs. However, adopting new hardware technology to improve dependability (such as installing a fault tolerant system) can be accomplished with AXP and VAX systems in a completely transparent fashion.

There are many levels of availability that are attainable with AXP and VAX configurations. This chapter discusses the major types of AXP and VAX system configurations from the hardware perspective. It addresses the following topics:

- Eliminating single points of failure. See Section 6.1.
- Conventional AXP and VAX systems. See Section 6.2.
- Fault tolerant VAXft systems. See Section 6.3.
- VMScluster hardware topologies. See Section 6.4.
- Dependability characteristics summary. See Section 6.5.

6.1 Eliminating Single Points of Failure

To eliminate single points of failure, *eliminate single points*. Use the methodology in the following list to discover single points of failure in your system.

1. Sketch a complete diagram of your system's configuration:
    - Show each significant subsystem separately, such as individual CPUs, disk drives, magtape controllers, and communications equipment.

- Use your service vendors to verify the accuracy of your diagram.

2. On the completed diagram, highlight those specific components used by:

   - All system users

     These components are the linchpins on which the entire system depends. Examples might include the paging and swapping disks, the main memory, the multiport terminal server, and the CPU.

   - The most important users

     These components are key to the mission-critical applications of your system. Examples might include a vector processor, the facsimile interface, and the disks where the quality database resides.

   - A multitude of users

     These components are important because their failure has widespread impact. Examples might include large disks that contain the directory trees of many users and a laser printer for Postscript files.

   - Only a few users

     These components may be present in your system's configuration incidentally and have little current use. Examples might include letter quality printers, printing terminals, and smaller removable disk packs.

   - No known users

     These components may be the result of changes in system usage during its lifetime. Examples might include card readers, older communications interfaces, plus even newer specialized options such as vector processors whose users have left the system.

   The point of this evaluation is to sort out the critical parts of your configuration from the unimportant ones. One potential action you could take as a result of this exercise is to remove the unused or rarely used subsystems from your configuration. That would alleviate not only their maintenance fees but also the possibility of their failure, which could impact the rest of the system.

3. Make a five-column list of contingency actions that could address the full or partial failure of each listed component as currently configured. Note that this list is likely to be long.

   - Order the list by first listing all users, then important users, and so on.
   - See Figure 6–1 for a suggested format for your list.

4. For each contingency action listed, note the consequences of any failure (that is, data loss, time loss, efficiency reduction, other consequences). Highlight those failures that have secondary consequences (for example, having to reenter transactions due to data loss).

5. With the help of your service vendors, complete the column that gives, for each appropriate component, a potential enhancement that would:

   - Prevent the failure from occurring

     Examples might include battery backup for main memory and plastic covers for device buttons.

   - Tolerate the failure when it occurs

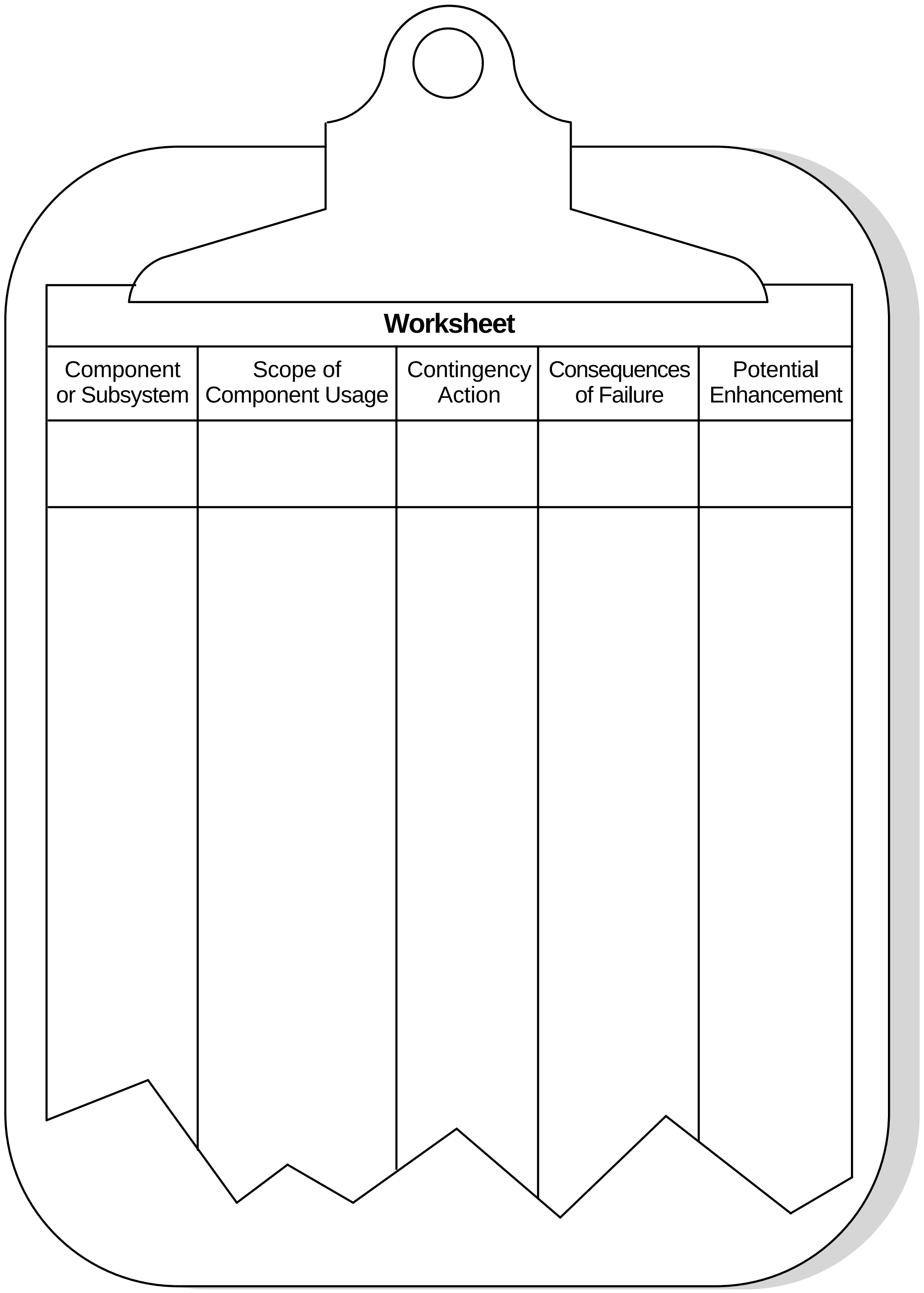
Examples might include shadowing a disk drive, installing an additional Ethernet controller, and utilizing error-correcting modems.

- Recover from the failure more effectively

  Examples might include adding an extra magnetic tape controller and drive, installing a second Postscript laser printer, and stocking critical spare parts on site.

6. Select those potential enhancements that:
   - Affect all the users
   - Affect the most important users
   - Alleviate serious secondary consequences
   - Fit your cost and other constraints
7. List these enhancements in the worksheet shown in Figure 2–3 according to which system users would benefit. Also list the enhancements in the worksheet shown in Figure 6–1, according to the impact of the enhancement on the computing system itself.

Figure 6–1 System Configuration Enhancements Worksheet



Worksheet

Component or Subsystem	Scope of Component Usage	Contingency Action	Consequences of Failure	Potential Enhancement

ZK−3689A−GE

6.2 Conventional AXP and VAX Systems

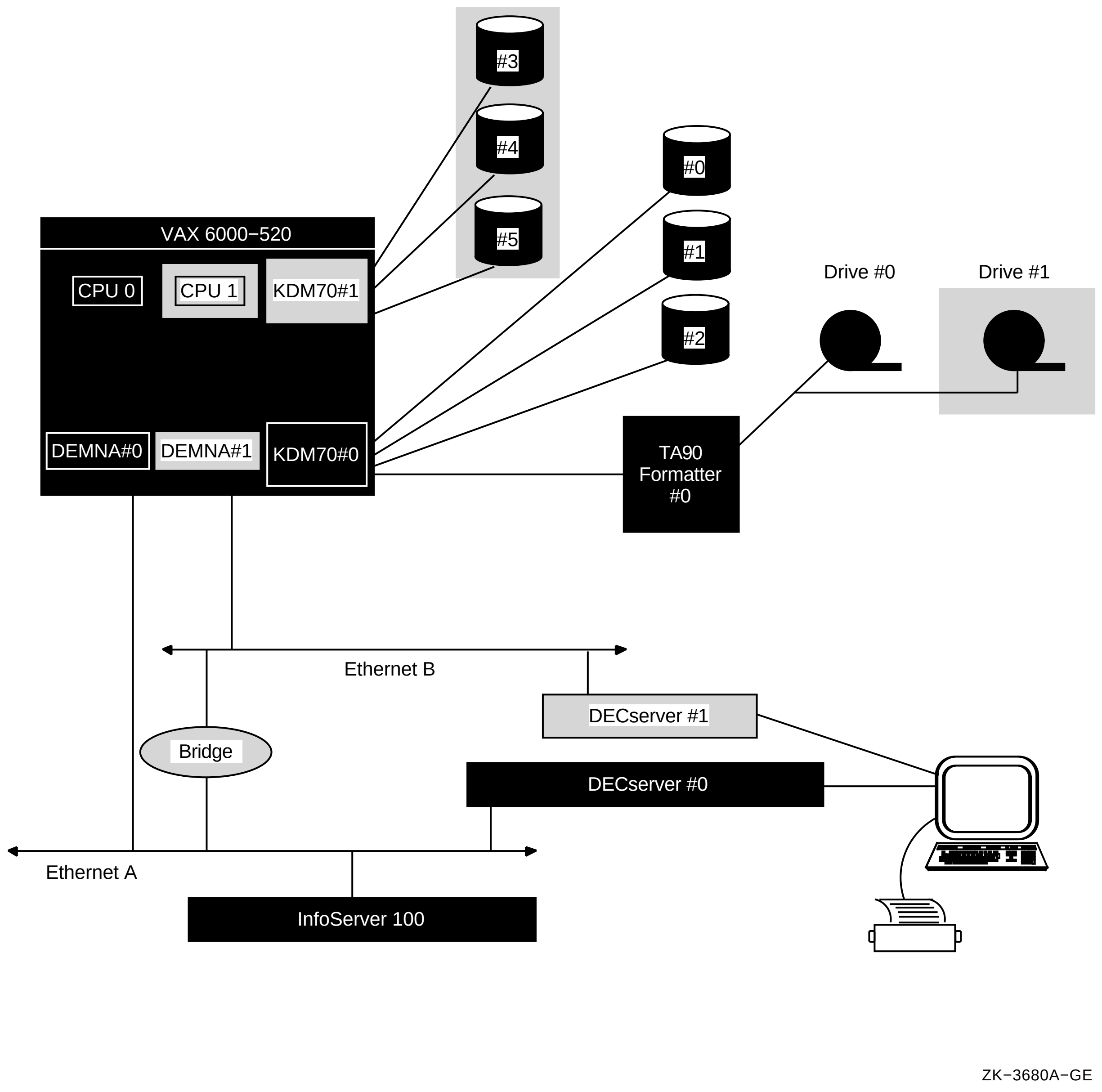
Conventional AXP and VAX systems are those system kernels that have not been included into VMScluster configurations. They may or may not be part of a network such as DECnet for OpenVMS or TCP/IP. **Conventional** also refers to hardware that, while engineered to be reliable, does not have the extensive built-in redundancy and instant, transparent recovery capabilities of a truly fault tolerant system, such as a VAXft system.

However, as described in the following sections, there are many ways to configure significant levels of redundancy into conventional AXP and VAX systems.

6.2.1 Dependability Enhancements to a Sample Configuration

The discussion in this section applies the methodologies described in Section 6.1 to the sample configuration shown in Figure 6–2. The shaded components in Figure 6–2 represent the additional pieces of hardware selected from the list of potential enhancements in Table 6–1. Ignoring the gray-shaded components yields a “before picture” of the system. Considering the gray-shaded components yields an “after picture” of the system.

Figure 6–2 Conventional System: Example Configuration



The configuration in Figure 6–2 is analyzed in Table 6–1, which lists potential enhancements to the system configuration. Again, the shaded components in Figure 6–2 represent the additional pieces of hardware selected from this table.

Table 6–1 System Configuration Enhancements: Sample Worksheet

Component or Subsystem	Scope of Component Usage	Contingency Action	Consequences of Failure	Potential Enhancement
CPU #0	All	Call service; reboot	System failure	Add CPU
Memory subsystem	All	Call service	System failure	Add battery backup
XMI™ backplane	All	Call service	System failure	None possible
KDM70 #0	All	Call service	System failure	Add KDM70
DEMNA #0	All	Call service	System not accessible	Add DEMNA
Ethernet A	All	Call service	System not accessible	Dual-rail Ethernet
InfoServer 100 #0	Some	Call service	Online documents not accessible	Add InfoServer 100
DECserver 200 #0	Most	Call service; use console	Cannot access system	Add DECserver 200
TA90 formatter #0	All	Call service	Cannot backup data; cannot exchange data	Add TA90 formatter
TA90 drive #0	All	Call service	Cannot backup data; cannot exchange data	Add TA90 drive
RA92 #0 (system disk)	All	Call service; restore backup	Cannot boot or run system	Shadow system disk
RA92 #1 & #2 (user disks)	Some on each	Call service; restore backup	Users cannot use system; data may be lost	Shadow user disks

To simplify the illustration, assume that cost and other constraints are not applicable. Consider the impact of implementing several key dependability enhancements. The shaded areas in the diagram indicate these enhancements (components added to the original configuration).

In Figure 6–2 when you consider the shaded components, note that many single points of failure have been eliminated. Even if a CPU failure caused a system crash, adding the second CPU would allow computing to immediately continue after a reboot, rather than having to wait for your service vendor to repair the CPU. Other failures (like the failure of the DEMNA Ethernet adapter) initiate a system or network reconfiguration that happens automatically and transparently to applications. Adding RA92 disk drives #3 to #5 allows both the system disk and the user disks to be shadowed, another automatic and transparent recovery capability. The shadow sets are not shown on the diagram but the new disks could be paired with the existing disks as #0 and #3, #1 and #4, #2 and #5.

Some failures (like a failure of TA90 drive #0) would still require operator actions to recover, while some failures like the memory subsystem cannot be tolerated at all and would require your service vendor to intervene. However, some subsystems that are single points of failure are also extremely reliable, like the memory subsystem or the XMI system backplane. Most Ethernet installations are also extremely reliable, but if your cables must pass through hazardous environments (such as some manufacturing plants), it might be wise to configure redundant Ethernet cables. That would require installing the second DEMNA Ethernet I/O adapter as shown in Figure 6–2. Another justification for installing a redundant Ethernet I/O adapter is if the computing system is acting as a host to personal computers (PCs) or as a boot member to satellites on the Ethernet. The PCs' or satellites' communication to their server node would then have some built-in redundancy.

If the terminal in the diagram represents the control terminal of a mission-critical manufacturing process, it may be worth the investment to also have redundant terminal servers—each connecting to one of the two asynchronous ports on most Digital terminals—or perhaps to two separate terminals at the control station. The shaded component labeled Bridge was added when the second Ethernet was installed. Network configuration rules required its installation between the segments.

Just how many of these enhancements would make sense in your context depends on your dependability requirements and other constraints.

An alternative method for achieving such a high level of redundancy is to connect two AXP systems or two VAX systems (or both an AXP and a VAX system) together to form a VMScluster system. This approach eliminates some key vulnerabilities of the enhanced configuration. If the memory subsystem of one system kernel failed, the VMScluster continues. If the operating system (or other privileged software) caused one system kernel to fail, the VMScluster continues. Figure 6–5, which appears later in this chapter, illustrates how such a VMScluster is configured.

A much easier method of utilizing a system with no single points of failure is to start with a system designed and built that way from the start. Section 6.3 discusses such fault tolerant configurations.

6.3 Fault Tolerant VAXft Systems (VAX Only)

VAX

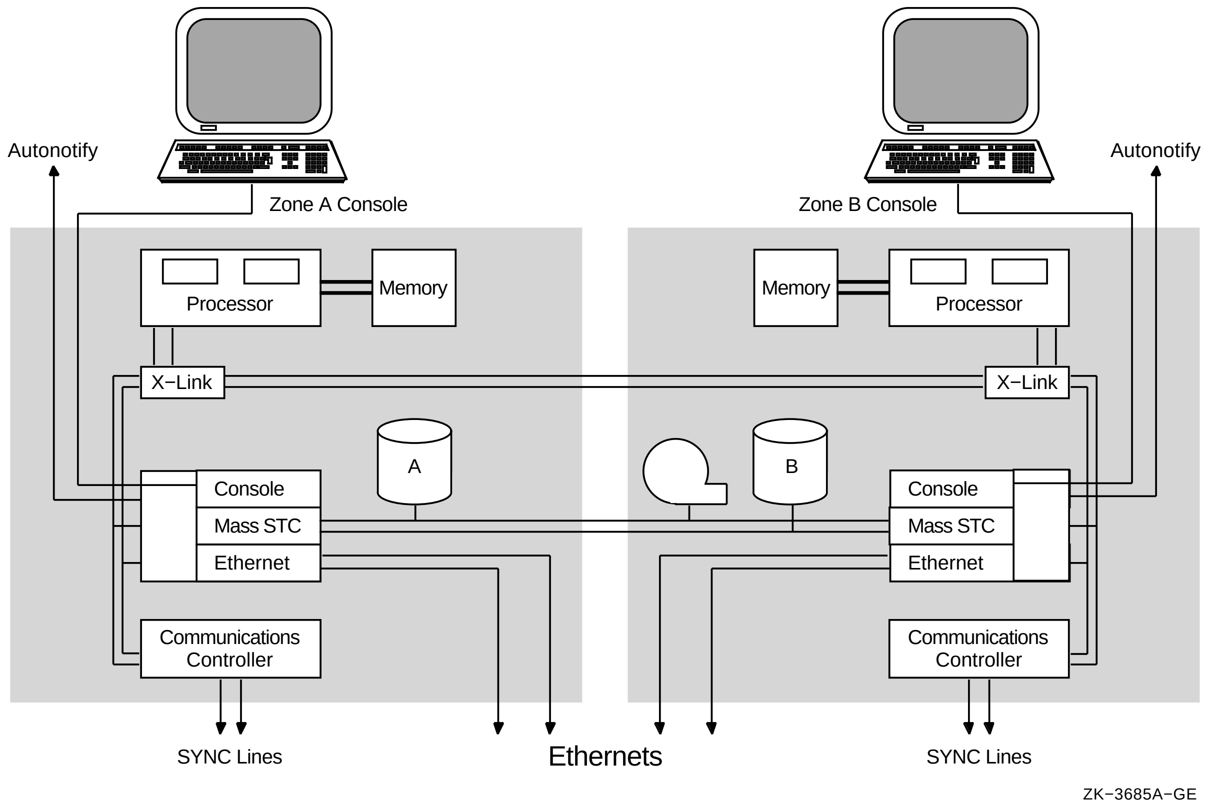
VAXft hardware configurations have no single point of failure so your dependability analysis process is greatly simplified.

Some of the potential contingency actions (such as stocking spare parts locally) would still be applicable so producing the list is still a useful exercise. The appropriate actions would tend to be more software and operations oriented, however, because contingencies related to hardware failures are transparently carried out by the VAXft system.

See Section 9.10 for more information on software related configuration issues. Even though your VAXft may not be part of a VAXcluster system, you can still use many VAXcluster oriented operating system features, such as Volume Shadowing for OpenVMS.

Figure 6–3 shows how the two redundant zones of a VAXft system work in parallel.

Figure 6–3 Fault Tolerant VAX System: Example Configuration



Note that in the case of a VAXft fault tolerant configuration, the only hardware concern is having twice as many disk drives as normally required. This allows shadowing of all the disks.

Because the two zones of a VAXft configuration are in physically separate cabinets (indicated by the shaded areas in Figure 6–3), your service vendor can perform repairs or replacements of all the subsystems in one zone while the other continues to provide service.

There is also no change in the CPU, memory, or I/O capacity of the system when a failure occurs. The applications and users do not detect the failure, but the system operators are notified via the error log and console terminals.

Note that the cables connecting the mass storage controller subsystems is a special DSSI that allows the independent repair of the zones. The cables connecting the X-Link subsystems enable the two zones to stay in lock step with one another. This is how the surviving zone can continue processing without a delay for catching up; it is always caught up.

If two VAXft system kernels were connected via DSSI, they would form a VAXcluster similar to the configuration shown in Figure 6–5. Like the VAXft, such a configuration would not suffer any loss of capacity when a hardware subsystem failed. It also would not be vulnerable to the failure of the building power. Not shown on the diagram are the built-in uninterruptible power systems (UPS) in each of the VAXft cabinets. As a VAXcluster, with independent copies of the OpenVMS operating system, the DSSI-based VAXft VAXcluster could continue to provide service even if a defect in privileged software caused the system to fail (or crash).

However, as a VAXcluster, each node would still momentarily pause during VAXcluster state transitions. These short pauses of a few seconds occur during significant configuration perturbations, such as a node entering or leaving the VAXcluster system. This time domain behavior must be taken into consideration for some critical applications that require fault tolerance. Of course, VAXft systems that are not members of a VAXcluster are not vulnerable to these pauses.

The Autonotify lines enable the console subsystems to dial out on a Data Phone® for immediate zone failure notification. ♦

6.4 VMScluster Hardware Topologies

There are thousands of ways to configure VMScluster systems. This handbook provides a brief overview of the general types of high performance and high availability configurations that are possible using VMScluster technology. See *VMScluster Systems for OpenVMS* for more details on how to configure, to set up, and to operate VMSclusters. Also refer to *Guidelines for VAXcluster System Configurations* for more detailed discussions about high availability configurations.

The following sections address this flexibility by focusing on the interconnect technologies. The interconnects discussed are those that the VMScluster software uses for Systems Communications Architecture (SCA) I/O traffic. This is different from standard network or terminal I/O traffic (which may or may not use the same interconnect simultaneously). A key OpenVMS subsystem that uses SCA connections is the OpenVMS distributed lock manager. By providing synchronizing services to the multiple copies of OpenVMS that execute on the AXP and VAX systems, the OpenVMS distributed lock manager is the linchpin that holds a VMScluster system together as a single, distributed computing system with transparently coordinated data access.

Because considerable disk and tape I/O traffic also shares the SCA interconnect with the OpenVMS distributed lock manager, the bandwidth and recovery capabilities of that interconnect have an impact on the configuration limits of the VMScluster system utilizing it.

Because VMScluster systems have more than one system kernel, you could apply the analysis methodology outlined in Section 6.2 to each of the system kernels and to the collections of disk, tape, and other peripherals that are common to all system kernels.

Note

For a large VMScluster system, such an undertaking could produce an exceedingly long list of components but the exercise is well worth it to identify critical single points of failure, none of which need to exist in a VMScluster system.

6.4.1 Ethernet Interconnect (IEEE 802.3)

The Ethernet Interconnect (IEEE 802.3) is the lowest cost method of configuring a VMScluster system. At 10 megabits per second, the Ethernet has sufficient bandwidth for several **boot members**[1] of significant capacity plus quite a few **satellite members**.[2] The practical limits on the number and size of boot members and satellite members depend on their usage. This is a situation where planning ahead in collaboration with your hardware and software vendors is critical. See Chapter 10 for more information.

Besides consulting experts who are familiar with the specifications of your Ethernet and computing equipment (and are also knowledgeable of current technology), you should also consider the following general guidelines:

- If possible, configure small disk drives into each satellite node. By placing local paging and swapping files on those disks, the I/O load on the Ethernet is reduced without losing the VMScluster management of having VMScluster common system disks.

  Note

  In a VMScluster, the AXP systems use an AXP system disk, while the VAX systems use a separate VAX system disk.

- Ensure that each satellite node has sufficient memory for its intended usage.
- Only the latest Ethernet adapters can sustain the full bandwidth of the cable. Because the adapter is frequently the bottleneck that limits the number of satellites that can be served by a particular boot member, adding additional adapters can alleviate the problem while providing redundancy.
- Ethernet cable redundancy can be achieved by laying two or more cables side-by-side and connecting every AXP or VAX node to both cables. Careful placement of Ethernet bridges and routers is required in this multirail configuration.
- Be sure to follow the documented guidelines for setting up a local area VMScluster. See *VMScluster Systems for OpenVMS* for more information on VMScluster system management.
- Refer to Section 9.10.3 for more information on rolling upgrades in VMSclusters with satellite nodes.

VMScluster system configurations typically use an Ethernet for network communications and terminal and printer connections. The types of VMScluster systems outlined in the following sections have the performance advantage of utilizing a separate interconnect for SCA I/O (disk, tape, lock manager) traffic.

Note

You should not expect as much I/O throughput from Ethernet-only VMScluster systems as from configurations that have separate SCA

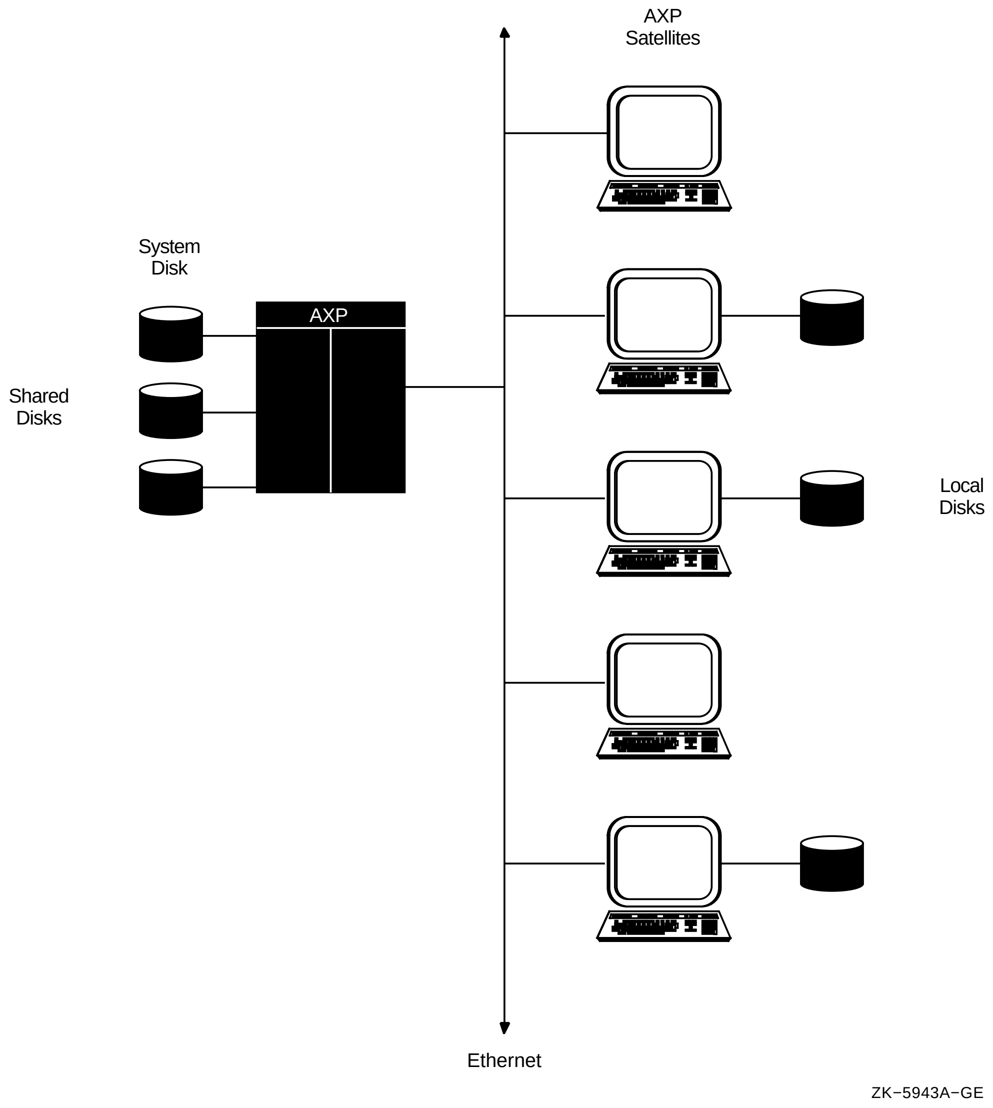
[1] AXP or VAX nodes with OpenVMS system disks that remotely "boot" the same-architecture satellite nodes.

[2] AXP or VAX nodes that use boot members as virtual system disks for their operation. Satellites may or may not have locally attached disks. AXP satellites use an AXP server to boot. VAX satellites use a VAX server to boot.

interconnects.

Figure 6–4 shows a local area VMScluster configuration.

Figure 6–4 Local Area (Ethernet) VMScluster System: Example Configuration



It might be advisable to shadow some key disks in this configuration, but the critical component of the VMScluster shown in Figure 6–4 is the AXP system. As the only boot member, if it fails, the satellites on the Ethernet also fail. You can gain a major advantage in total system dependability by adding a second AXP or VAX processor to form a DSSI-based VMScluster.

6.4.2 Digital Storage Systems Interconnect (DSSI)

The Digital Storage Systems Interconnect (DSSI) is an interconnect that AXP and VAX systems can use to access disk and tape peripherals. Each peripheral is an integrated storage element (ISE) that contains its own controller and mass storage control protocol (MSCP™) server that works in parallel with the other ISEs on the DSSI. With a bandwidth greater than 4 megabytes (40 megabits) per second, the interconnect also supports the direct connection of up to four systems. When four systems are used with DSSI, no more than two AXP systems can be part of the configuration; that is, you can have one of the following combinations:

- Four VAX systems
- Three VAX systems and one AXP system
- Two VAX systems and two AXP systems

A DSSI VMScluster system with one AXP node and one VAX node is shown in Figure 6–5.

The combination of high throughput and common data access by the AXP and VAX nodes provides a way to configure VMScluster systems at much lower cost than using the CI computer interconnect and HSC controllers, but with much greater capacity than using only the Ethernet (IEEE 802.3) Interconnect.

Because both nodes have direct access to their own system disks, DSSI VMScluster systems can support the OpenVMS rolling upgrades. This capability allows the version of OpenVMS to be upgraded on one node and that node to be rebooted while the other node is still providing service.

General guidelines for DSSI VMScluster systems follow:

- Currently, no more than four nodes may be connected to DSSI.
- Multiple DSSI paths can be supported by most AXP and VAX system kernels. This allows the total amount of storage in your system to be increased significantly.
- Some DSSI adapters (for example: KFQSA) do not support the SCA protocol. Therefore, those adapters can only be used to serve their disks to other nodes.
- Consult your Digital support representative for more complete and timely configuration details on DSSI VMScluster systems.

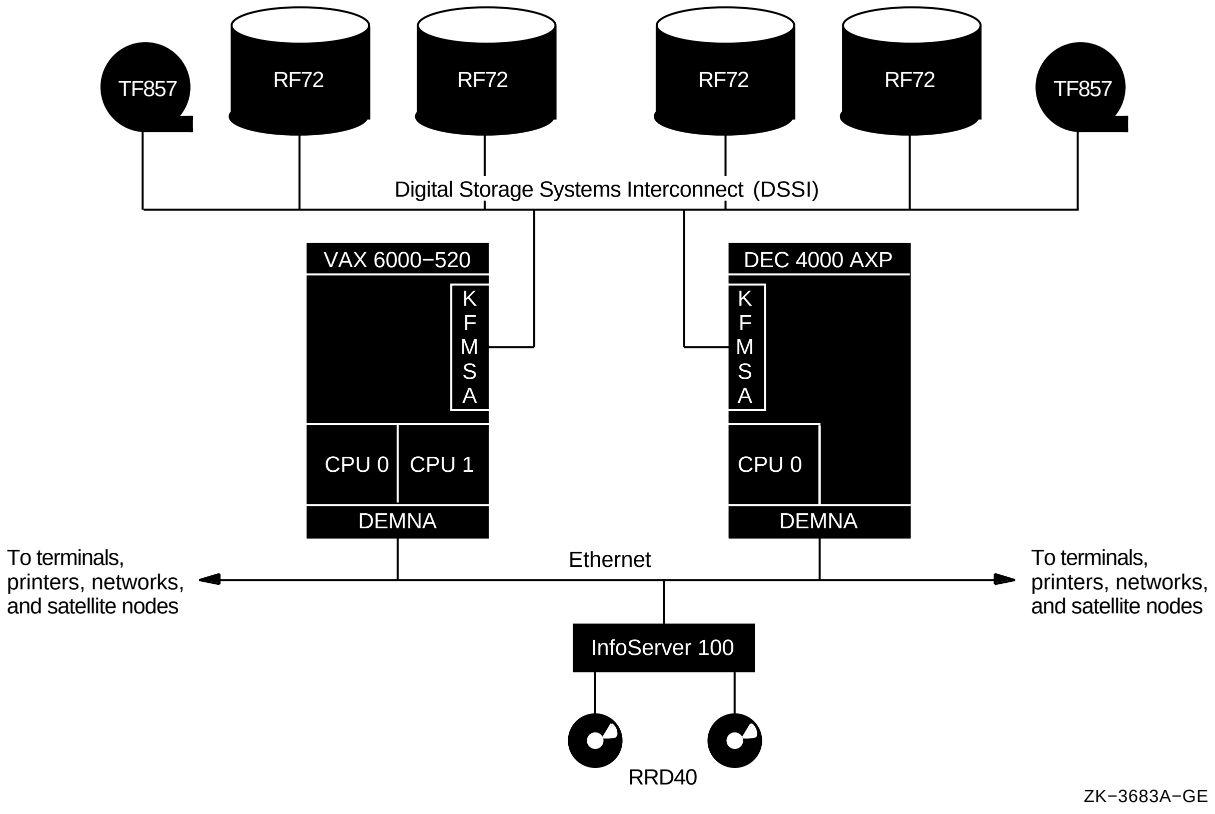
Note

As its name Digital Storage Systems Interconnect (DSSI) implies, DSSI has been optimized as a high bandwidth path to peripheral devices. One feature designed into its architecture is the ability to add or remove ISEs without disturbing normal system operation. There are configurations where conventional AXP or VAX nodes connected to DSSI cannot be added or removed in a likewise online fashion. This is because the termination circuitry of DSSI cables might be removed when a system kernel is disconnected from the cable. This means that in some cases, repairing one system kernel in a DSSI VMScluster would require the shutting down and rebooting of all the system kernels. Contrast with CI VMSclusters, Section 6.4.3.

However, VAXft systems use special connections between their zones to circumvent these restrictions so the zones of a VAXft system may be serviced independently.

Figure 6–5 shows a DSSI VAXcluster configuration.

Figure 6–5 DSSI VMScluster System: Example Configuration


TF857
RF72
RF72
RF72
RF72
TF857
Digital Storage Systems Interconnect (DSSI)
VAX 6000−520
KFMSA
CPU 0
CPU 1
DEMNA
DEC 4000 AXP
KFMSA
CPU 0
DEMNA
To terminals, printers, networks, and satellite nodes
Ethernet
To terminals, printers, networks, and satellite nodes
InfoServer 100
RRD40
ZK−3683A−GE

This DSSI-based VMScluster configuration already has few single points of failure. It would be important to shadow the key disks, which could require the purchase of additional drives. There are several scalar CPUs for adequate capacity, although if adequate performance after a failure is critical, it would be advisable to install a “spare” or two.

Assume that the VAX 6000 shown in Figure 6–5 already has a vector processor. If the applications that require speed are important enough to the success of the business, it would make sense to add a second vector processor on the VAX 6000. (AXP systems do not support vectors.) If serving workstations or personal computers on the Ethernet is a critical business function of the system, then adding second Ethernet adapters (DEMNAs) to the VAX 6000 system and the DEC 4000 AXP system would provide redundancy plus additional throughput capacity.

6.4.3 CI Computer Interconnect

The CI is a high performance, fault tolerant way to connect AXP and VAX nodes to disk and tape storage devices and to each other. To implement its star topology, it uses redundant coaxial cables that operate at a 70 megabit per second data rate. Because two transmit and two receive cables form a single CI physical connection, if one cable fails, the others continue to provide service. Star couplers enable all VAX nodes and the HSC disk and tape controllers to communicate directly. The star couplers operate at full cable bandwidth and are also dual redundant (2*n - see Chapter 2).

The HSC intelligent disk and tape controllers also connect to star couplers via the CI cables. Because the DIGITAL Storage Architecture (DSA) disk and tape drives have dual access ports, you can configure HSCs in a redundant fashion. By having two HSCs connected to each disk or tape, if one of the HSCs fails, the other continues to provide service. Failover support in the OpenVMS operating system makes this capability transparent to application programs (except for a short pause in I/O processing during failover).

General guidelines for CI based VMScluster systems follow:

- Some AXP to CI, or VAX to CI, adapter subsystems cannot support the full throughput of the CI cable. If your AXP node or VAX node is a very powerful model (such as a DEC 7000 or a VAX 10000), you may need to use the latest CI adapters (such as the CIXCD) and to install more than one CI adapter per AXP or VAX. The latter option provides increased throughput via CI port load sharing feature of the OpenVMS operating system and also provides some redundancy. An AXP node or a VAX node that is fully operational except for its CI adapter may as well be powered off because it cannot participate as part of the VMScluster system.
- Another reason to install more than one CI adapter to an AXP node or a VAX node in your VMScluster system is to support more than one star coupler. Multiple star couplers may operate in a VMScluster configuration if they each have at least one AXP or VAX node attached. The maximum number of AXP nodes and VAX nodes in a VMScluster configuration is not increased by having multiple star couplers. However, the maximum number of HSC or HSJ controllers is dramatically increased, allowing much more storage to be configured into the system.
- You should attempt to spread the VMScluster disk I/O traffic between several drives, and if possible, spread the most active disk drives between HSC controllers. This will reduce the peak I/O load that must be processed by any one HSC. Note that when a disk or tape drive is connected to redundant HSCs, at any single point in time only one of the HSCs is controlling the device. For example, Drive 17 can receive its I/O from either HSC A or HSC B, but not both at once. If HSC B is controlling Drive 17 and HSC B fails, then HSC A will continue with no loss of data or context.
- You are not, however, restricted to fully utilizing one HSC or HSJ device while the other HSC or HSJ device sits idle. From the set of disks and tapes controlled by a redundant HSC pair or HSJ pair, you can assign some drives to be controlled by—for example—HSC A and others to HSC B. This will spread the I/O load between the HSCs under normal operating conditions. See the *OpenVMS I/O User's Reference Manual* for details about the set preferred path feature of the OpenVMS I/O subsystem.

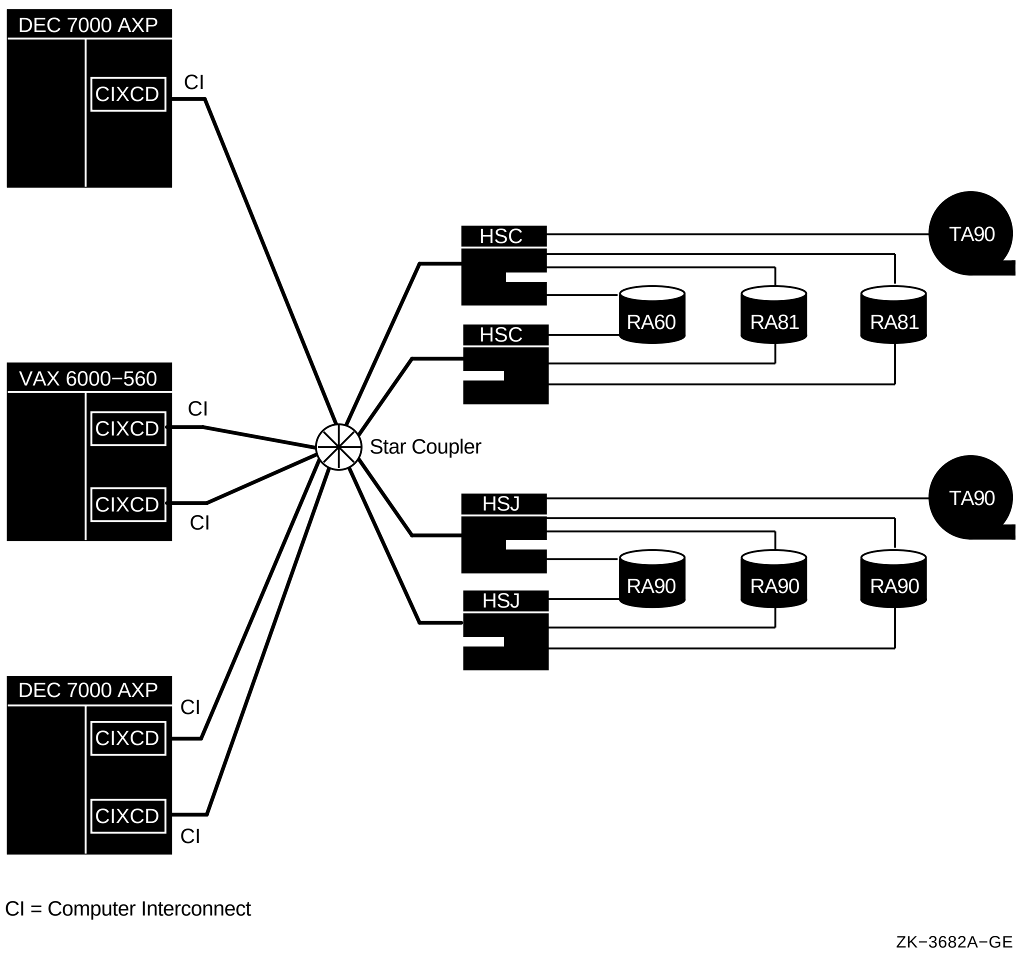
Note

Look in SYS$EXAMPLES:PREFER*.* for a Macro™ programming example that demonstrates the set preferred path feature. The sample program is provided by the OpenVMS operating system. (SYS$EXAMPLES on VAX systems also includes a BLISS version of the preferred path program sample.)

- Disk drives that do not need to be cluster common (such as those with a node's page and swap files) can be directly attached to their AXP nodes and VAX nodes via controllers like the Digital KDM70. This will help reduce the total I/O traffic through the CI adapters, on the CI cables, and through the HSC or HSJ controllers. In addition, consider direct attachment of solid state disks (such as the Digital ESE-20) to powerful nodes as a way to alleviate I/O bottlenecks that are caused by frequent disk head seek operations.

Figure 6–6 shows a CI VMScluster configuration.

Figure 6–6 CI VAXcluster System: Example Configuration



The primary areas to address with this fairly redundant configuration in Figure 6–6 would be environmental. Ensuring sufficient (and spare) air conditioning capacity and clean, preferredly uninterruptible, electric power are two immediate concerns. If sufficient disk drives are present to shadow key disks, the next consideration is CPU capacity after a failure.

Although all three nodes in this CI-based VMScluster system are quite powerful system kernels, the two DEC 7000 AXP systems supply most of the total CPU capacity. If one of the DEC 7000 AXP systems should fail, would your applications or users be able to tolerate the resulting degradation in response times? Also note that the top DEC 7000 AXP system in Figure 6–6 has only one CIXCD adapter. While that interface could likely handle the I/O load that the single CPU system can generate, if it failed, that entire DEC 7000 AXP system would become unavailable for use.

As stated in Chapter 2, you should also consider the amount of load carried by a surviving subsystem when its twin fails.

What appears to be a single point of failure in Figure 6–6 is the star coupler that connects all the CI lines. All AXP and VAX systems in any type of VMScluster must have direct connections to all other nodes. The CI is a fully redundant communications medium. Each CIXCD (or any other CI adapter) connects to four coaxial cables. There are two for transmitting and two for receiving. The full CI bandwidth (70 megabits per second) is available over either the A path or the B path.

The star coupler is not a single point of failure because there are actually two star couplers in every box: one for the A path cables and one for the B path cables. Star couplers are also immune to power failures because they contain no powered components but are constructed as sets of high frequency pulse transformers. Because they do no processing or buffering, star couplers are not I/O throughput bottlenecks. They operate at the full rated speed of the CI cables. However, in very heavy I/O situations, it is possible to overload the CI cable/coupler system and to require multiple star couplers.

Most configurations use multiple star couplers to increase the number of AXP, VAX, HSC, and HSJ nodes that can be connected. The CIXCD model CI adapters have the throughput advantage of being able to use both the A path and the B path simultaneously when both paths are functioning, as is usually the case.

6.4.4 Fiber Distributed Data Interface (FDDI)

The Fiber Distributed Data Interface (FDDI) is an interconnect that uses fiber-optic cables for high bandwidth over long distances. The data rate is 100 megabits per second and the range is 200 kilometers.

Because FDDI uses a token-ring topology with two counter-rotating rings, the actual maximum distance between any two lobes of an FDDI VMScluster system is approximately 40 kilometers. A **lobe** is a collection of AXP nodes, VAX nodes, or both connected by a CI and star coupler or connected by a DSSI. Such a collection could exist on its own as a VMScluster system, but when more than one lobe is connected by the FDDI, it becomes one large, multiple site data center that comprises a VMScluster system. Figure 6–7 shows how two lobes can be connected into one VMScluster system with FDDI.

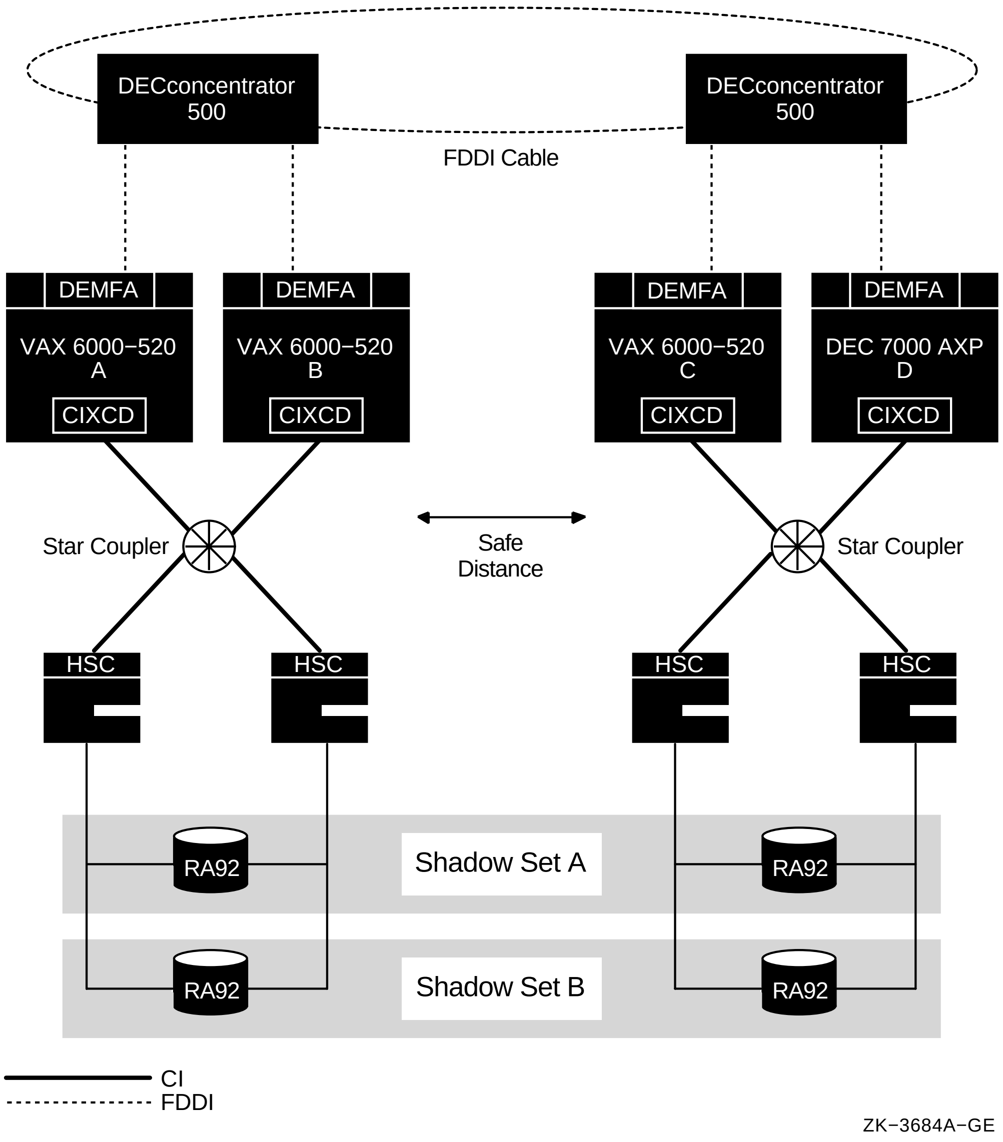
If the lobes of an FDDI VMScluster system are located in different buildings or even different cities, the configuration exhibits a level of disaster tolerance. If Volume Shadowing for OpenVMS is used to ensure that key files or databases at both lobes are kept in synch, a catastrophic environmental failure at one physical site (such as a fire) would not prevent the entire VMScluster system from delivering continuous service.

Because of the potentially long distances that must be covered by the I/O traffic in an FDDI VMScluster system, extra care is required in the setup of OpenVMS system parameters. Refer to *Managing the Business Recovery Server* and *Configuring the Business Recovery Server* for details about configuring, setting up, and operating FDDI VMScluster systems using the Business Recovery Server software product.

See *VMScluster Systems for OpenVMS* and *Guidelines for VAXcluster System Configurations* for related VMScluster configuration information.

Figure 6–7 shows an FDDI-based VMScluster configuration.

Figure 6–7 FDDI VMScluster System: Example Configuration



In Figure 6–7, the elliptical line connecting the two DECconcentrator 500 units represents the dual-direction FDDI cables. Nodes VAX A and VAX B make up one lobe of the VMScluster. Nodes VAX C and VAX D make up a second lobe. The FDDI fiber cable connects both lobes into a single VMScluster system.

Although this FDDI based VMScluster system is configured to be almost completely immune to environmental failures (assuming an uninterruptible power system (UPS) at each site), special care must be taken to consider all parts of the system. If, for example, both sites were served by the same telephone central office, the access to the computing system by remote terminal users could be disrupted by a failure of this hidden single point of failure.

Note that Volume Shadowing for OpenVMS is used to maintain key data storage devices in identical states (shadow sets A and B). Any sectors (on the shadowed disks) written at one site will also be written (virtually simultaneously) at the other site, and vice versa. Database transaction semantics provide additional robustness to ensure that critical data items at both sites are identical on a continuing basis.

See Chapter 5 for more information about ways to make your computing system safe from environmental failures.

6.4.5 Mixed Interconnect VMSclusters

A mixed interconnect VMScluster is any VMScluster system that uses more than one type of interconnect for SCA traffic. Some large configurations may use several cable segments of the interconnect. These multisegment connections are explained in Chapter 7. If there is only one type of interconnect, it is not a mixed interconnect VMScluster.

Examples of mixed interconnect VMScluster configurations are:

- A CI-based VMScluster with satellite workstations on the Ethernet
- A dual-host server (DSSI VMScluster) with satellites on the Ethernet
- A CI-based VMScluster where one of the VAX nodes connects to another VAX node via the DSSI, or one of the AXP nodes connects to another AXP node via the DSSI
- Any FDDI based, multilobe VMScluster

Table 6–2 compares the characteristics of the different VMScluster interconnect technologies. Note that the capability to use multiple cable segments and multiple star couplers (together with multiple different interconnects) provides the flexibility to configure high I/O capacity systems.

Table 6–2 VMScluster Interconnect Characteristics Summary

Interconnect	Transfer Rate (megabits/second)	Maximum Diameter (meters)
Ethernet (coaxial)	10[1]	2,800
Ethernet (microwave)	10	35,600
Ethernet (fiber-optic cable)	10	60,000
DSSI	40	19[2]
CI	70[3]	90
FDDI	100	40,000

[1]Per IEEE 802.3 specification, newer adapters (such as the DEMNA) are required to achieve the full rated bandwidth of the cable.

[2]This distance depends on the particular ISE and AXP or VAX models in the configuration. Consult your Digital representative for your context.

[3]Per path, the CIXCD adapter can achieve even higher throughput by using both CI paths simultaneously.

Note

VMScluster systems operate as single computing systems. The timing considerations (propagation delays and bandwidths) of VMScluster protocols (Systems Communication Architecture, or SCA) place tighter constraints on interconnect configurations than would more general networking protocols (DECnet for OpenVMS).

Note that some early dual-host server configurations (where there is no DSSI in the configuration) are not mixed interconnect VMSclusters. By using only the Ethernet for SCA, these configurations are in fact local area VMSclusters.

General guidelines for mixed interconnect VMScluster systems follow:

- All mixed interconnect VMScluster configurations require a common network interconnect (either Ethernet or FDDI).
- Some especially large and complex mixed interconnect VMScluster configurations may require a special service contract for the most effective support.
- Refer to Section 9.10.3 for more information on rolling upgrades in VMSclusters with satellite nodes.

6.5 Dependability Characteristics Summary

So far, the figures in this chapter have illustrated some example configurations as a way to help you decide which configuration best meets your dependability requirements. This section helps you compare different configurations with VMScluster topologies in wide area networks (WANs). Figure 6–8 illustrates an example configuration of a WAN.

Figure 6–8 Wide Area Network: Example Configuration

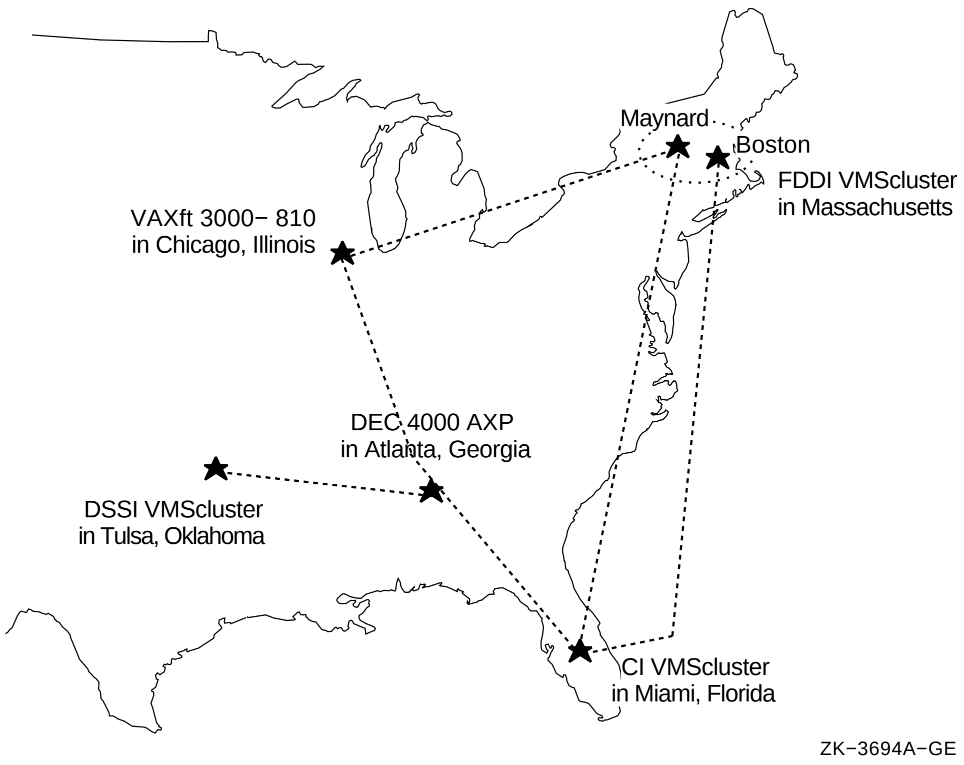

Maynard
Boston
FDDI VMScluster in Massachusetts
VAXft 3000− 810 in Chicago, Illinois
DEC 4000 AXP in Atlanta, Georgia
DSSI VMScluster in Tulsa, Oklahoma
CI VMScluster in Miami, Florida
ZK−3694A−GE

Figure 6–8 and the other figures in this chapter are compared in Table 6–3. The last column in the table gives the characteristics of WANs to enable you to compare them with VMScluster topologies. Note that the VMScluster in Tulsa would lose contact with the rest of the network if the node in Atlanta failed. See Chapter 7 for details on the dependability characteristics of networks.

Consider using the Table 6–3 format to identify which dependability features are required for your particular context and to see which hardware configurations have those capabilities. If more than one configuration meets your needs, the final selection could be made with other factors in mind, such as cost, currently installed equipment, and distances.

Table 6–3 Dependability Characteristics Summary

Dependability Feature	Conventional VAX or AXP	Fault Tolerant VAX	Local Area (Ethernet) VMScluster	DSSI VMScluster	CI VMScluster	FDDI VMScluster	Mixed Interconnect VMScluster	WAN[4]
Instantaneous failure/recovery		X	X[1]	X[1]				
Single security domain	X	X	X	X	X	X	X	
Disaster tolerant						X	X	X
Transparent data access	X	X	X	X	X	X	X	
Supports Volume Shadowing for OpenVMS	X	X	X	X	X	X	X	
VMScluster DECnet™ alias			X	X	X	X	X	
Login/batch load balancing			X	X	X	X	X	
Very high I/O bandwidth	X	X		X	X	X	X	
Rolling upgrades		X		X	X	X[2]	X[2]	X
Kernel repair during production		X	X		X	X	X	X
CPU capacity unchanged by kernel failure		X	X[1]	X[1]				
Survives power failure		X	X[1]	X[1]		X[3]	X[3]	X[3]
Managed as a single system	X	X	X	X	X	X	X	
More than three system kernels			X		X	X	X	X
Survives operating system failure			X	X	X	X	X	X

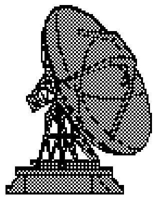
[1]When comprised exclusively of VAXft systems.

[2]Applies only to boot nodes.

[3]If this configuration spans independent power grids.

[4]Wide Area Network (WAN).

7 Dependability Characteristics of Communications Networks



In the simplest terms, a dependable communications network exists when your design ensures that data can be transmitted reliably from its current location (point A) to its destination (point B) in a timely manner, especially when that data delivery is critical to your business. Fault-free networking is a fundamental contributor to the success of your business applications. Using dependable equipment and appropriate levels of data path redundancy is an effective solution for supporting critical networking applications. Network dependability requires sufficient performance so that users and applications do not have to wait for service. Network dependability also requires resilience. For wide area host-to-host communications and for local area networks, resilience requires fault tolerance with regard to data corruption and dynamic alternative routing in the event of line or server failures. Taking advantage of dynamic alternative routing involves your planning in setting up redundant paths or the ability to create a redundant path using switching techniques.

Equipment reliability is also an important part of network dependability, as are vendor service levels and the ability to properly manage your networking software and hardware components. After you have eliminated all *single points of failure* for the networking connections that will support mission-critical applications, you very likely will want to use software monitoring and failure analysis tools to quickly identify the location and name of any failed hardware components. Even redundant networking configurations can fail, and you should implement proactive strategies for finding and repairing failed components promptly.

This chapter summarizes a variety of networking hardware products, redundant configurations, and software products that enhance the dependability of your network. You are directed to the primary Digital product documentation sets for details about the products' capabilities and performance considerations. This chapter and the information presented in the referenced documents can help you develop strategies for meeting the networking needs of your business.

Topics described in this chapter:

- The varying degrees of protection that can be implemented for networks. See Section 7.1.
- The ability to provide multiple paths in local area networks (LANs) from an AXP or a VAX node to Ethernet segments, Fiber Distributed Data Interface (FDDI) fiber-optic cable, or both. See Section 7.2.

- A local area VMScluster failure analysis program provided by OpenVMS to support your network dependability failure recovery strategies. See Section 7.3.
- The ability to use FDDI as a VMScluster *interconnect*. See Section 7.4.
- The ability to provide highly available connections from a fault tolerant VAXft system to public and private wide area networks (WANs) and to local synchronous devices. See Section 7.5.
- The availability impact of the DECnet cluster alias name in networking applications that access VMScluster systems. See Section 7.6.
- VAX Distributed Name Service (DNS) software, which lets you establish control over the way applications reference network resources. DNS provides a consistent, network-wide set of names for distributed applications. By using DNS, network applications are able to continue processing when the physical location of named network resources change. See Section 7.7 for information about DNS.
- VAX Distributed File Service (DFS) software, which provides DECnet for OpenVMS users with transparent access to files stored on remote OpenVMS disks as if the files were stored on local disks. See Section 7.8 for information about DFS.
- Other products that support network dependability goals including DECalert, NMCC/VAX ETHERnim, LAN Traffic Monitor, and NMCC/DECnet Monitor. See Section 7.9 for introductory information about these tools. Refer to Section 11.6 for information about a Digital Customer Support Center (CSC) data center that invented DECalert and used the other networking tools to meet CSC requirements for 100% application availability.

Note

In February 1994, Digital granted Target Systems Corporation a nonexclusive, nontransferable license to use DECalert Version 1.2 software. Under the agreement, Target Systems may provide support for the Digital customer base (existing and future) for the product, as well as future development and migration of the product or its derivatives. Target Systems named the product TARGET–>ALERT.™ Digital reserves the right to continue to sell DECalert or its enhancements, as Digital deems fit.

The software fault tolerance features provided by the Digital Reliable Transaction Router (RTR) are described in Section 8.1. Also see Section 10.13 for information about networking consulting services from Digital.

7.1 Degrees of Protections from Networking Faults

Networking solutions provide varying degrees of protection from faults. The solutions can be selected according to the criticality of the application.

At the communication-line level, parity checks and overlaying protocols provide tolerance of intermittent and soft faults for synchronous and some asynchronous lines. Redundancy and triangulation can be used to provide alternate paths in the event of line failure. Telecommunications companies automatically provide this protection for leased lines. Only the line between your organizations and the lines to any leased lines are vulnerable.

Ethernet and Fiber Distributed Data Interface (FDDI) LANs can be configured in a segmented manner, so that a failure in one area will not bring down the entire network. Ethernet and FDDI LANs also can be configured in multirail mode to provide a redundant LAN segment as protection against a failure of a communications adapter, a LAN segment, or both.

Terminal servers, which connect terminals to a host node in a LAN, offer enhanced availability by handling different numbers and varieties of lines. Each terminal connected to a given server can access services on any other line connected through that server. In a VAXcluster, terminal servers increase system availability by providing login load balancing and automatic session failover. In the latter case, terminals logically connected to a failed host can access other hosts on the LAN.

Application availability is increased by the server's ability to establish a logical connection to any local service node that implements the LAT™ protocol. This means that users can connect to all the services offered (rather than to the node itself). Servers also protect the network against unauthorized disruption by providing multiple levels of security.

Ethernet LANs can be extended by using bridges that manage traffic between two or more local LANs. Bridges provide a substantial boost to availability through their automatic configuring and self-initializing capabilities. Bridges can be arranged in any topology, enabling them to be used in parallel with other bridges to provide an automatic backup when needed. Bridges that cause loops in this situation automatically enter a backup state, monitoring traffic and taking over in the event of a failure to the primary bridge.

The following list presents a general approach to achieving network dependability:

- Evaluate the proposed design and identify single points of failure. Draw an X through any connection and consider what is affected when the network is unavailable to the users and applications that rely on that connection.
- Eliminate single points of failure (where possible and affordable) by building in redundant paths to the components. If providing redundant paths does not appear to be affordable, weigh the cost of the hardware redundancy versus the business impact (cost) of losing the users and applications that depend on the unavailable components.
- Do everything you can to minimize risks. Identify and document any remaining risks, and develop contingency plans to be implemented when (not if) the risk turns into a failure.
- Set the Network Control Program (NCP) or the Network Command Language (NCL) parameters to match the actual primary use of a line. NCP and NCL are DECnet tools.
- For wide area network (WAN) connections established to transmit voice and data transmissions, compare the costs of operating:
  - Separate voice and data networks
  - A combined voice and data network over T1 lines provided by a telephone carrier

  Once you approximate the relative costs, weigh each option with the dependability requirements of the applications and of the users relying on the connections.

- Identify limitations in your network availability and design any distributed applications with these limitations in mind.
- Take advantage of product features that enhance network dependability. Examples include:
  - Using additional LAN adapters to an AXP or VAX node to ensure that a network connection continues to exist in the event that one adapter fails, one LAN segment fails, or both. OpenVMS provides support for this capability. The additional adapters can connect the node to Ethernet LAN segments or to FDDI LAN segments or rings. For each adapter, define a line and circuit to make the adapter available to DECnet for OpenVMS or LAT. See Section 7.2.

Note

When using the LAN adapters for DECnet for OpenVMS, each LAN adapter must be connected to a separate extended LAN because DECnet for OpenVMS changes the network address used by the LAN adapter.

  - Duplicating terminal servers in an Ethernet LAN environment to provide redundancy for users on terminals with dual ports. This step secures user access to system resources in the event that one of the Ethernet LANs fails. Each user should have two communication lines available to each terminal; that is, one line to each terminal server on each Ethernet LAN.

    Terminal servers connect asynchronous terminals at speeds up to 19.2Kb per second. See the *Networks Buyer's Guide* for details. This Digital publication is released every six months.
  - Customizing a local area VMScluster failure analysis program called LAVC$FAILURE_ANALYSIS so that you can quickly identify faulty components in your network by using device names and descriptions that you provide. The source code for this Macro program is provided by OpenVMS in your system's SYS$EXAMPLES directory. The LAVC$FAILURE_ANALYSIS program is summarized in Section 7.3 and detailed in *VMScluster Systems for OpenVMS*.
  - Using the higher bandwidth and length of FDDI fiber-optic cable, where appropriate and possible.
  - Using the cluster alias name instead of individual node names for remote network connections to server VMScluster systems, especially in distributed applications.

The remainder of this chapter describes features that enhance network dependability.

7.2 Providing Multiple Paths to Ethernet and FDDI Segments

OpenVMS supports the use of multiple communications adapters (up to four with each VAX node, and up to two with each AXP node. This support provides automatic failover of your network connection in the event that one adapter fails, or one LAN segment fails, or both. For the latest configuration details, refer to the VMScluster Software for OpenVMS *Software Product Description* (SPD 42.18.*xx*).

The DEC FDDIcontroller 400 communications controller (also called DEMFA) provides the FDDI LAN interface. While Ethernet meets the requirements of many applications, the ten-fold increased speed of FDDI fiber-optic cable may be used to support network-intensive applications.

Table 7–1 lists the supported adapters.

Table 7–1 Ethernet and FDDI Adapters

Bus	Ethernet Adapters	FDDI Adapter
XMI	DEMNA	DEMFA
BI	DEBNA, DEBNI	
Q–bus	DELQA, DESQA, DEQTA (DELQA-YM)	
UNIBUS™	DEUNA™, DELUA™	
Integral	LANCE, SGEC	

At boot time, all Ethernet and FDDI adapters are automatically configured for local area VMScluster use. PEDRIVER automatically detects and creates a new channel for each unique pair of LAN adapters between the local node and each remote cluster node. Channel viability is continuously monitored every three seconds at a minimum. Channel failure does not interfere with node-to-node (virtual circuit) communications as long as there is at least one remaining channel functioning between the nodes.

Note

An AXP or VAX node with dual communications adapters can operate in this manner with a DECnet for OpenVMS end-node license. In other words, it is not required that you have a full-function DECnet for OpenVMS license to use the multiadapter feature.

7.2.1 Recommendations for High Availability of Local Area VMSclusters

Follow these general guidelines to create highly available local area VMSclusters:

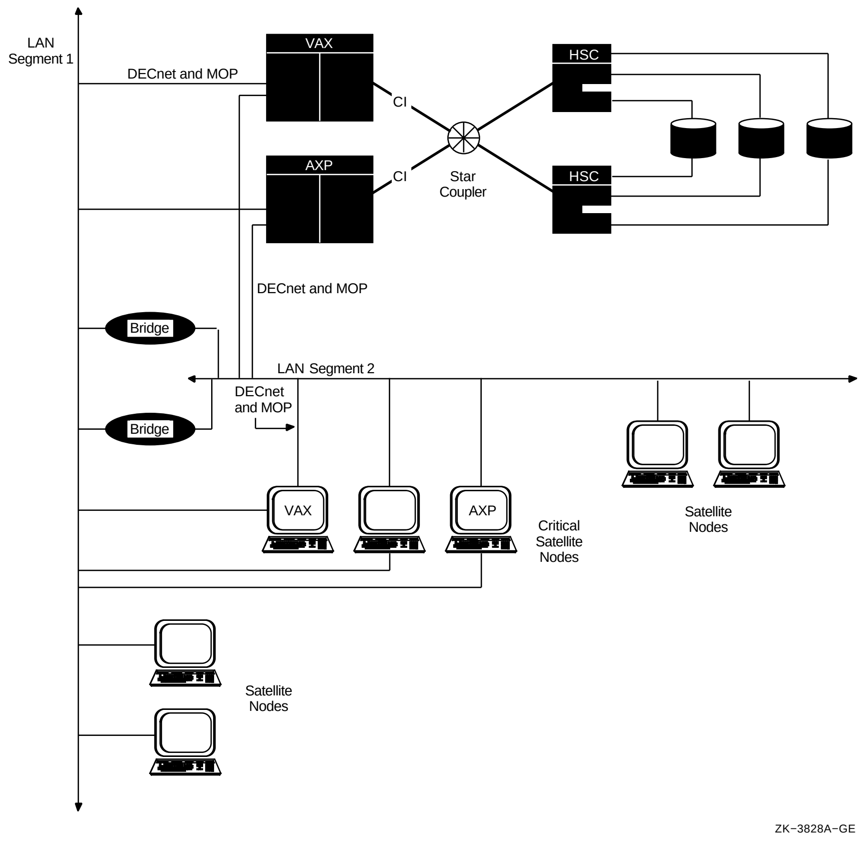
- Bridge VMScluster LAN segments together to form a single extended LAN.
- Provide redundant LAN segment bridges for failover support.
- Configure LAN bridges to pass the local area VMScluster and Maintenance Operations Protocol (MOP) multicast messages.

VMScluster Systems for OpenVMS describes the minimum requirements, plus more detailed configuration guidelines, for local area VMScluster systems that use multiple adapters. If you follow the guidelines, server nodes (nodes serving disks and lock traffic) can typically use some of the additional bandwidth provided by the added LAN adapters and increase the overall performance of the VMScluster. However, the performance increase depends on the configuration of your VMScluster and the applications it supports. See *VMScluster Systems for OpenVMS* for details.

7.2.2 Sample Local Area VMScluster Configurations with Multiadapter Connections to LAN Segments

Figure 7–1 shows a sample configuration for local area VMScluster systems connected to two different LAN segments.

Figure 7–1 Sample Configuration for a Local Area VMScluster Connected to Two LAN Segments

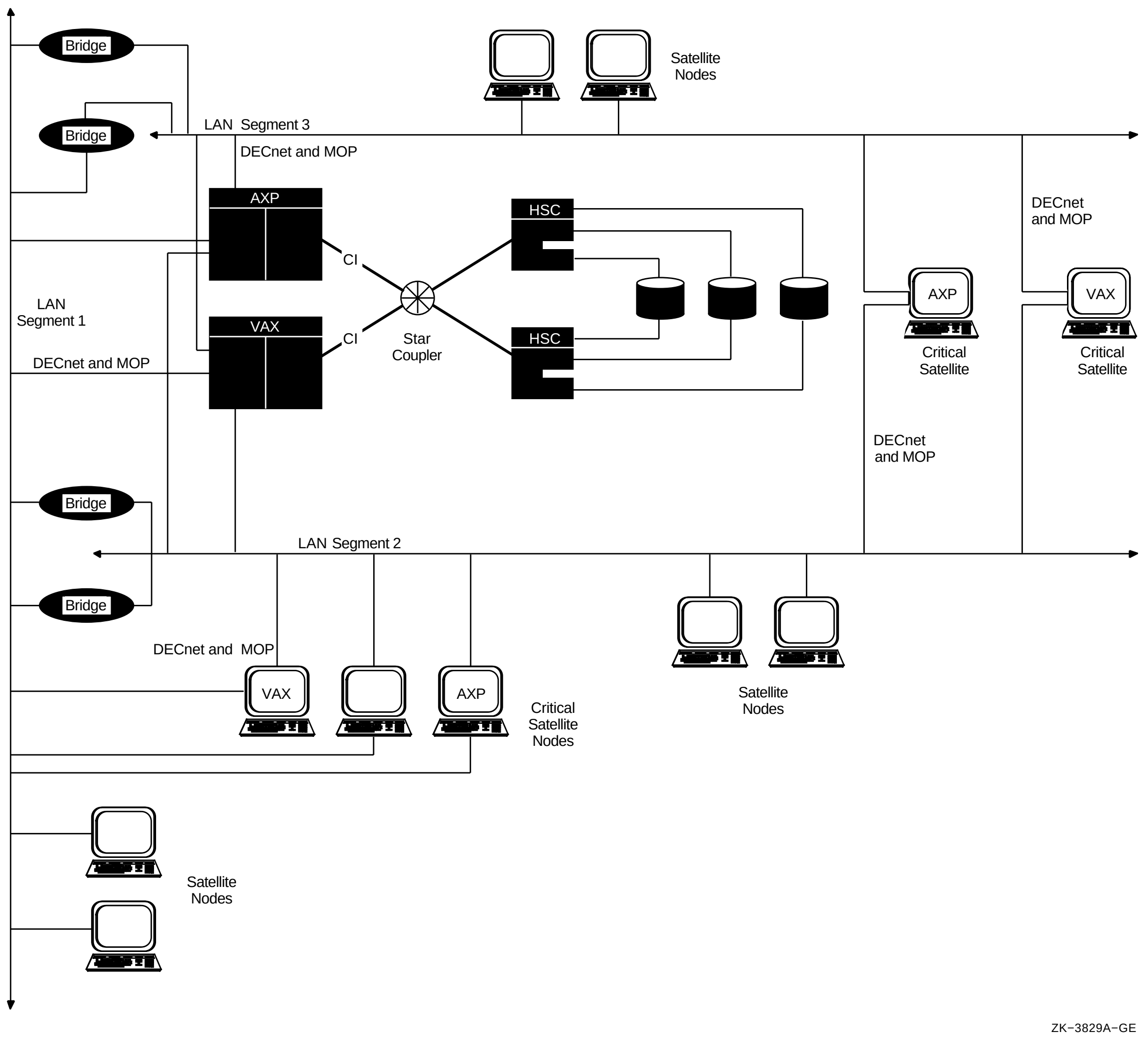


In Figure 7–1:

- Connecting critical nodes to multiple LAN segments provides increased availability in the event of segment or adapter failure. The disk server nodes in the CI VMScluster can use some of the network bandwidth provided by the additional network connection. (CI is a computer interconnect.) Critical satellites can boot using the other LAN adapter if one LAN adapter fails.
- Connecting noncritical satellites to only one LAN segment helps to balance the network load by equally distributing systems among the LAN segments. These systems communicate with satellites on the other LAN segment through one of the bridges.
- Using redundant LAN bridges prevents the bridge from being a single point of failure. Any single bridge can fail and all nodes in the VMScluster will remain members of the VMScluster.
- Satellite node booting is ensured by providing a DECnet for OpenVMS Maintenance Operations Protocol (MOP) connection to each LAN segment. (See the *DECnet for OpenVMS Networking Manual* for details about MOP.)

Figure 7–2 shows a sample configuration for local area VMScluster systems connected to three different LAN segments.

Figure 7–2 Sample Configuration for Local Area Cluster Systems Connected to Three LAN Segments



In Figure 7–2:

- Connecting disk servers to two or three LAN segments can help provide higher availability and better I/O throughput.
- Connecting critical satellites to two or more LAN segments can also increase availability. If any of the network components fail, these satellites can use the other LAN adapters to boot and still have access to the critical disk servers.
- Distributing noncritical satellites equally among the LAN segments can help to balance the network load.
- Satellite node booting is ensured by providing a DECnet for OpenVMS Maintenance Operations Protocol (MOP) connection to each LAN segment. (See the *DECnet for OpenVMS Networking Manual* for details about MOP.)

7.2.3 Ethernet and FDDI Options

Properly configured Ethernet and FDDI connections can provide cost-effective communications within local area networks. Ethernet is used extensively today because it provides network managers with flexible, modular growth of networks; you can expand computing resources while preserving the investment in existing systems.

In evaluating new options posed by FDDI, define your goals:

- If your goal is networking performance, then FDDI alone is a reasonable choice. FDDI provides 100mb per second line speed, while Ethernet provides 10mb per second.
- If your goal is availability, then you should design a highly redundant network that will probably combine FDDI, CI, and Ethernet components.
- If you currently are using Ethernet in your local area network (LAN) and your goal is the growth of your LAN, determine by how much you expect it to grow. If the answer is that the network will double in size over the next 5 to 10 years, then multiple Ethernet segments should be fine. That is, if the expected growth is only two or three times the current level, then preserving the investment in Ethernet hardware and adding some new segments and LAN bridges should be less expensive than replacing the major network components with FDDI hardware.

  However, if the answer is that the network will increase 10-fold, then consider FDDI. (The bandwidth of FDDI is 10 times that of Ethernet, yet the cost of FDDI is less than 10 times that of Ethernet.)
- If your only goal is to minimize cost (without a second concern for availability), then Ethernet is probably best for your site. Note that FDDI components are between two and five times more expensive than Ethernet components.
- If your goal is disaster tolerance, you do not necessarily have to build geographically distant sites that span 40 kilometers (possible with FDDI configurations using the VMScluster Business Recovery Server software product). Some sites require disaster tolerance but on a more modest geographic scale. For instance, an extended Ethernet LAN between two buildings that are less than one mile apart might be adequate protection against a fire.

Note

The actual maximum distances that are possible with Ethernet are approximately 2 kilometers for a single Ethernet LAN and 10 kilometers for an extended LAN through four LAN bridges. For details, see the *Networks Buyer's Guide*, which is published by Digital every six months.

7.2.4 Allowing for LAN Bridge Failover

To achieve high availability, Digital recommends redundant bridges between LAN segments. If one bridge fails, another bridge can support message traffic between the LAN segments. To help ensure that there is little delay between the failure of one bridge and the continuing of message traffic support by another bridge, you should make sure the local area VMScluster follows these guidelines:

- Bridge timers should be set to be faster than VMScluster timers.
- Bridge self-test times should be less than the value for the system parameter RECNXINTERVAL.

See *VMScluster Systems for OpenVMS* for a detailed discussion about adjusting the VMScluster times and the RECNXINTERVAL system parameter.

7.2.5 Alternate Adapter Booting for Satellite Nodes

OpenVMS supports booting from any LAN adapter on a local area VMScluster satellite with multiple LAN adapters. You can boot from an alternate adapter to work around broken adapters or network problems. You can also use this feature to boot into different clusters, depending on the adapter you use to boot the system.

See *VMScluster Systems for OpenVMS* for information about booting from an alternate adapter.

7.2.6 Changing the LAN Address in the DECnet Database to Allow a Cluster Satellite to Boot with Any Adapter

DECnet for OpenVMS Phase IV supports one LAN hardware address per node definition. To allow a cluster satellite with multiple LAN adapters to use any LAN adapter to boot into the cluster, use one of the following methods:

- Define a synonym node with a different DECnet address.
- Create and maintain different DECnet databases on the different boot nodes within the VMScluster.

VMScluster Systems for OpenVMS explains your options in more detail.

7.3 Troubleshooting with the VMScluster Network Failure Analysis Program

In network design, providing a backup mechanism such as a second LAN adapter for a VAX node helps to ensure that a single point of failure has been eliminated. However, when the primary component fails and the backup component takes over the communications work, your network is again vulnerable to a single point of failure. For this reason, it is critical that you use tools that can quickly identify the precise name and location of failed components.

For local area VMScluster environments, OpenVMS provides the Macro source file of a failure analysis program. This program, LAVC$FAILURE_ANALYSIS, is available in your SYS$EXAMPLES directory.

You must tailor the LAVC$FAILURE_ANALYSIS.MAR file so that it includes a physical depiction of the local area VMScluster physical layout, plus information about each component. You then assemble and link the program and execute it on one or more VMScluster system(s) that will perform the failure analysis. When the program executes, it provides the physical description of your cluster communications network to the set of routines that perform the failure analysis.

Using the network failure analysis program can help reduce the time necessary for detecting and isolating a failing network component, therefore providing a significant increase in local area VMScluster availability. LAVC$FAILURE_ANALYSIS groups together channels that fail and compares them with the physical description of the VMScluster network. The program then develops a list of nonworking network components related to the failed channels and uses OPCOM messages to display the names of components that have a probability of causing one or more channel failures. If the network failure analysis cannot verify that a portion of a path (containing multiple components) works, the program calls out the first component in the path as the primary suspect. Other components are listed as secondary or additional suspects.

Sections 7.3.1 and 7.3.2 summarize the LAVC$FAILURE_ANALYSIS program and a related subroutine package for additional functions. See *VMScluster Systems for OpenVMS* for details on the local area VMScluster failure analysis program and the related subroutine package.

7.3.1 Summary of Using the Failure Analysis Program

Copy the LAVC$FAILURE_ANALYSIS.MAR source file from SYS$EXAMPLES to a private directory. With a text editor, create a diagram of your VMScluster communications network. When you edit LAVC$FAILURE_ANALYSIS.MAR, include this drawing in the program's source file. Your drawing should show the physical layout of the VMScluster and include the following components:

- LAN segments or rings
- LAN bridges
- Wiring concentrators, DELNI devices, or DEMPR™ devices
- LAN adapters
- VAX systems

For large clusters, it may be necessary to verify the configuration by tracing the cables.

Note

VMScluster Systems for OpenVMS includes a detailed example of how you could illustrate a sample network configuration. The example shows how the illustration is used by the LAVC$FAILURE_ANALYSIS program.

Give each component on the drawing a unique label. If your VMScluster contains a large number of nodes, you may want to replace each node name with a shorter abbreviation. For example, you can replace the node name ASTRA with A and call node ASTRA's two LAN adapters A1 and A2.

List the following information for each component:

- Unique label.
- Type [SYSTEM, LAN_ADP, DELNI].
- Location (the physical location of the component).
- LAN address or addresses (if applicable). (Devices such as DELNI devices, DEMPR devices, and cables do not have LAN addresses.)

Next, classify each component into one of the following categories:

- Node: VAX system in the VMScluster configuration.
- Adapter

  A LAN adapter on the VAX system that is normally used for VMScluster communications.
- Component

  A generic component in the network. Components in this category can usually be shown to be working if at least one path through them is working. Wiring concentrators, DELNI devices, DEMPR devices, LAN bridges, and LAN segments and rings typically fall into this category.
- Cloud

  A generic component in the network. Components in this category cannot be shown to be working even if one or more paths are shown to be working. This type of component is necessary only when multiple paths exist between two points within the network. An example is redundant bridging between LAN segments. At a high level, multiple paths can exist; however, during operation, this bridge configuration allows only one path to exist at one time. In general this bridge example is probably better handled by representing the active bridge in the description as a COMPONENT and ignoring the standby bridge. You can identify the active bridge with such network monitoring software as Remote Bridge Management System (RBMS) or Digital Extended LAN Management Software (DECelms). With the default bridge parameters, failure of the active bridge will be called out.

Use the component labels to describe each of the connections in the VMScluster communications network.

Finally, choose a node or group of nodes to run the network failure analysis program. You should run the network failure analysis program only on a node that you included in the physical description when you edited LAVC$FAILURE_ANALYSIS.MAR. The network failure analysis program on one node operates independently from other systems in the VMScluster. So, for executing the network failure analysis program, you should choose systems that are not normally shut down. Other good candidates for running the program are systems with the following characteristics:

- Faster CPU speed
- Larger amounts of memory
- More LAN adapters (running local area VMScluster protocol)

See *VMScluster Systems for OpenVMS* for detailed information about how to edit and use LAVC$FAILURE_ANALYSIS.MAR.

7.3.2 Summary of Subroutine Package

In addition to the LAVC$FAILURE_ANALYSIS program, OpenVMS places in the SYS$EXAMPLES directory:

- Two programs called LAVC$START_BUS.MAR and LAVC$STOP_BUS.MAR to start and stop the local area VMScluster protocol on a specified LAN adapter.

- A subroutine package to extend control the local area VMScluster programs LAVC$FAILURE_ANALYSIS.MAR, LAVC$START_BUS.MAR, and LAVC$STOP_BUS.MAR.

  The subroutine package includes function calls to control the network failure analysis program:

  - SYS$LAVC_DEFINE_NET_COMPONENT—Creates a representation of a physical network component
  - SYS$LAVC_DEFINE_NET_PATH—Creates a directed list of network components between two network nodes
  - SYS$LAVC_ENABLE_ANALYSIS—Enables the network failure analysis so that the future channel failures are analyzed
  - SYS$LAVC_DISABLE_ANALYSIS—Stops the network failure analysis and deallocates the memory used for the physical network description

See *VMScluster Systems for OpenVMS* for details about the programs and the subroutine package.

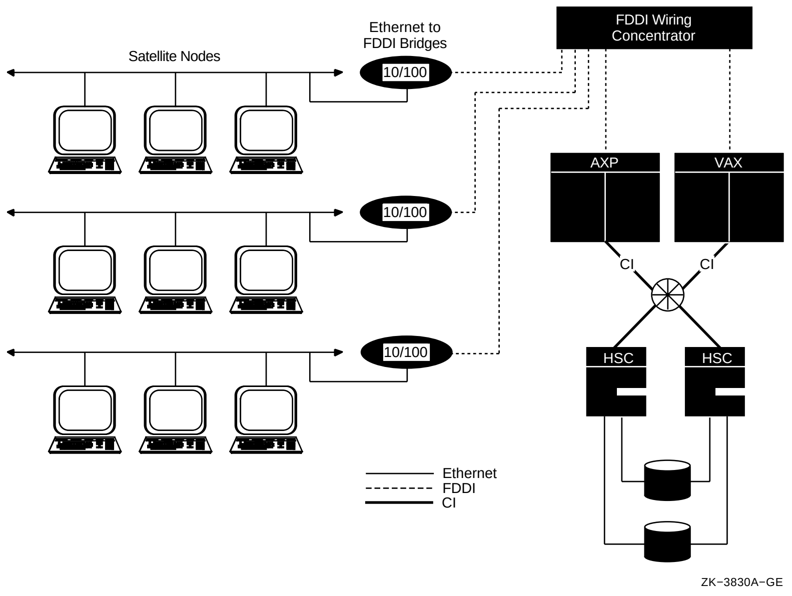
7.4 Network Configurations Using FDDI as the VMScluster Interconnect

Fiber Distributed Data Interface (FDDI) fiber-optic cable can provide distinct advantages for networked client/server applications. Desktop environments can be integrated into VMScluster systems by using FDDI's higher bandwidth (ten-fold speed improvement over Ethernet) and FDDI's ability to connect computing resources that are located up to 40 kilometers away.

Before FDDI, VAXcluster systems that were used for desktop integration typically had several large VAX processors acting as boot and disk servers, plus many satellite nodes (clients) taking advantage of the served resources. Ethernet is used in this configuration to connect the satellites to the servers. Typically, satellites (workstations) on multiple Ethernet segments are bridged (using multiple LAN Bridge 200 devices, also called DEBAM devices) into a DELNI Ethernet wiring concentrator. From the DELNI, connections are made to the boot and disk server nodes of the CI VMScluster.

VMScluster systems that have heavily used Ethernet segments can replace the Ethernet backbone with FDDI to eliminate the Ethernet as a performance bottleneck. As shown in Figure 7–3, FDDI can replace the Ethernet from the bridges to the server CPU nodes. This configuration can increase overall throughput.

Figure 7–3 FDDI in Conjunction with Ethernet in a VMScluster System



The longer distances provided by FDDI might permit you to create new VMScluster systems in your computing environment. For example, another way of looking at Figure 7–3 is that this VMScluster system is new. The large nodes on the right could have replaced server nodes that previously existed on the individual Ethernet segments.

Currently there are no storage controllers that connect directly to FDDI. Nodes connected to FDDI must have local storage or access to storage over another interconnect. In Figure 7–3 the AXP and VAX systems have CI connections for storage. If a VMScluster system has more than one FDDI-connected node, then those CPU nodes will probably use CI or DSSI connections for storage. The CI-connected AXP and VAX systems in Figure 7–3 are considered a lobe of the VMScluster system. A **lobe** is a set of CPUs that are connected together by one or more VMScluster interconnects. A lobe must have its own system disk(s), used by all CPUs in that lobe. A single CPU with a local system disk can be a lobe. Not all CPUs are part of a lobe, however. For example, a set of workstations is not a lobe.

For data center consolidation, FDDI can expand your data center to include additional resources while retaining a single system management domain. Computing resources and their associated storage that are today physically located outside your data center can now be tied in with FDDI; these resources could include standalone systems or another VMScluster system. When connecting two or more existing VMScluster systems with FDDI, you must create a single new VMScluster system. The previously independent VMScluster systems can remain physically separated within a building or in several buildings. But the systems now logically form a single operating environment, as depicted in Figure 7–4.

Figure 7–4 Multiple-Site Data Center VMScluster System

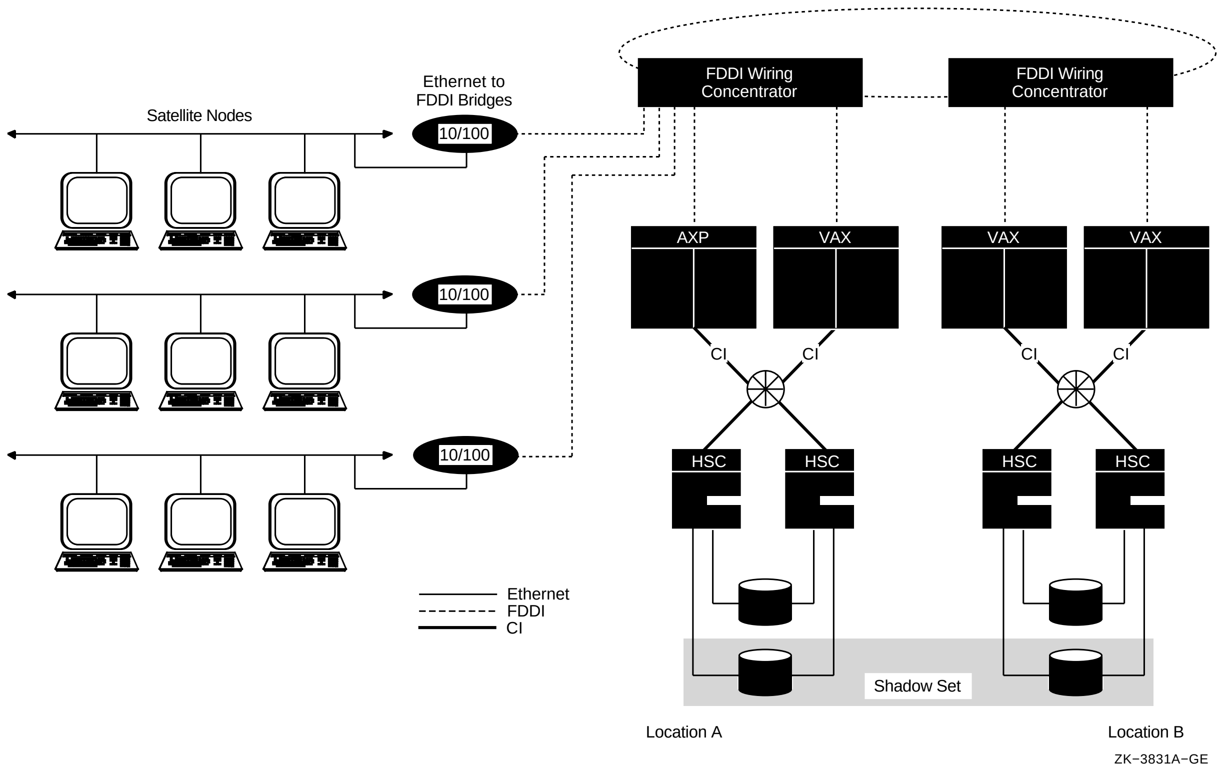


Figure 7–4 shows a multilobe VMScluster configuration. It builds on Figure 7–3 where one CI based VMScluster system was serving a large number of satellites. The AXP and VAX processors at *location A* and their CI connected storage have been identified as one lobe. If more compute power or storage resources are required, additional systems can be connected in through FDDI. At *location B*, the AXP and VAX processors and their CI connected storage comprise a second lobe. A lobe can also be a DSSI-connected VMScluster system or a single CPU node with a local system disk. The satellite nodes can now use the resources at location A (as they did before) plus the newly available resources at location B.

Refer to *Managing the Business Recovery Server* and *Configuring the Business Recovery Server* for details about configuring, setting up, and operating FDDI VMScluster systems using the Business Recovery Server software product. Also consult the *Networks Buyer's Guide* for configuration guidelines about FDDI networks. *VMScluster Systems for OpenVMS* and *Guidelines for VAXcluster System Configurations* cover related VMScluster information.

Note

Your Digital representative can obtain for you the current Business Recovery Server *Software Product Description* (SPD 35.05.*xx*) and VMScluster Software for OpenVMS *Software Product Description* (SPD 42.18.*xx*).

7.5 Providing Multiple WAN Connections for VAXft Systems (VAX Only)

VAX

The DEC WANcontroller 620 (DFS32) is a synchronous communications controller that provides highly available connections to public and private wide area networks (WANs) and to local synchronous devices. A fault tolerant VAXft system consists of two fully redundant processing zones, as described in Section 6.3 and as illustrated in Figure 6–3. Each zone of a VAXft system can support up to four DEC WANcontroller 620 communications controllers, with each controller supporting two synchronous lines.

This controller supports standard layered communications software for DECnet, VAX P.S.I., BISYNC, and OpenVMS SNA applications. The two lines are independently managed and operated, allowing different protocols to be run simultaneously. Each controller supports redundant connections to a single synchronous line; this feature maintains communications integrity during VAXft zone servicing and failover due to a faulty controller. ♦

7.6 Using a DECnet Cluster Alias to Promote Network Application Availability

The VMScluster manager should define a cluster alias name for the VMScluster. Programmers of client network applications that use the services of a VMScluster should use the alias name. Doing so ensures that the remote access will be successful when at least one VMScluster member is available to process the client program's requests. Networked requests of client programs that use an individual VMScluster node name will fail whenever that particular VMScluster node is unavailable.

Because a VMScluster can be used as a single computing system, the DECnet for OpenVMS networking software can be configured to take special action in that context. When enabled, the DECnet cluster alias makes all the VMScluster nodes appear to be one node, from the point of view of the rest of the network. This helps other nodes establish communications to your VMScluster system more reliably because the connection is not dependent upon a particular node being reachable as long as a single node in the VMScluster is reachable.

The nodes in a VMScluster do not require DECnet for OpenVMS routing to be enabled for the VMScluster to operate. But, if you choose to use a VMScluster DECnet cluster alias, you must enable routing on at least one of your nodes. For greater dependability of the alias function in your VMScluster, more than one node should enable routing and set up the alias. Then, the DECnet cluster alias will not disappear if the routing node becomes unreachable.

Currently, the DECnet for OpenVMS software cannot perform the alias function for more than 63 nodes in a VMScluster configuration. Refer to *VMScluster Systems for OpenVMS* for information on setting up a VMScluster DECnet cluster alias.

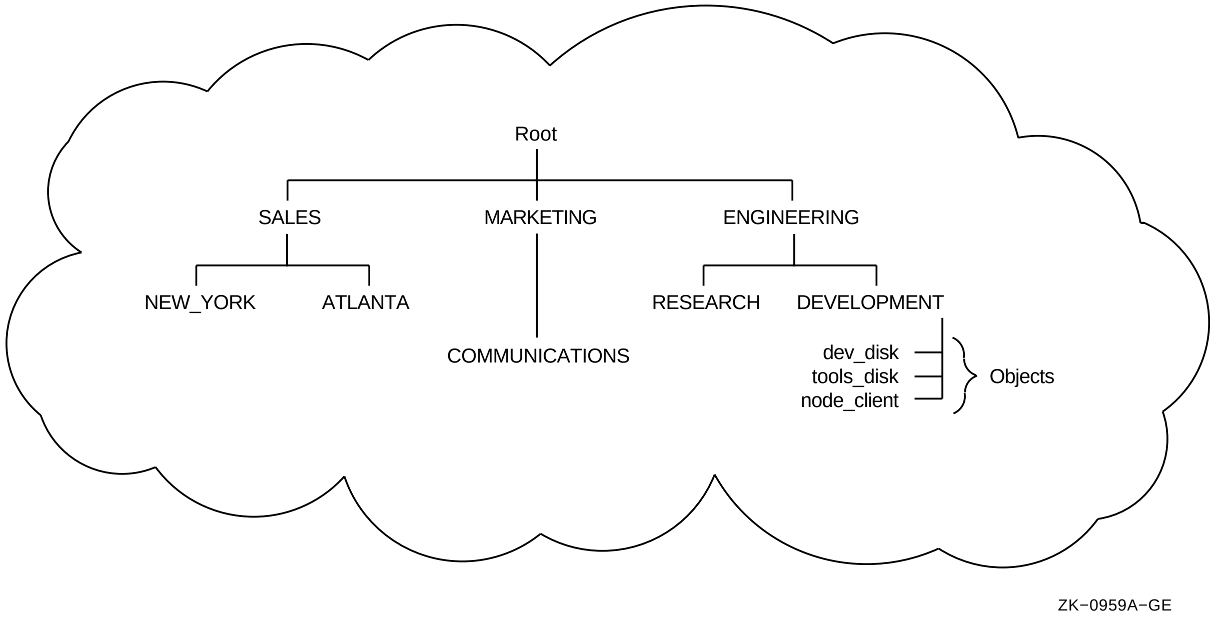
7.7 Using DNS to Support Network Dependability (VAX Only)

VAX

VAX Distributed Name Service (DNS) lets you assign unique names to network resources. Once a network application has named a resource using DNS, the name is available for all users of the application. Users can move from one system to the next and refer to application resources by the same name. This characteristic is significant because over the lifetime of a network application, it is likely that some or many resources will be moved to a different location. The dependability goals of network applications that use DNS are enhanced because the applications are better able to withstand changes in the location of resources.

The collection of names in the DNS database is called a **namespace**, as depicted in Figure 7–5.

Figure 7–5 A DNS Namespace



A namespace is located on OpenVMS nodes where the DNS server software is installed. At the top of the hierarchy is the root directory. Below the root directory are levels of subdirectories. The namespace administrator establishes the directory structure of the namespace and, in some cases, assigns names to directories. While the organization of the namespace is similar to the OpenVMS directory structure, namespace directories are completely separate from the OpenVMS file system's directory structure.

The DNS software refers to the named resources in the namespace as **objects**. Each object refers to a specific entry. Associated with each object is a set of attributes describing properties about the object. An application reads object attributes for information such as a network node address or object status.

The DNS clerk, which is located on every OpenVMS VAX system, receives application requests from the DNS clerk system service. This system service lets an application register a resource in the namespace and then access the resource from any point in the network by a single name. All applications designed to take advantage of the DNS software must use either the DNS clerk system service or the DNS run-time routines to register, modify, or find information in the DNS database.

See the *VAX Distributed Name Service Management* manual for details about using DNS in network applications. ♦

7.8 Using DFS for Transparent Network File Access (VAX Only)

VAX

VAX Distributed File Service (DFS) provides DECnet for OpenVMS users with transparent access to files stored on remote OpenVMS disks as if they were on local disks. DFS assists network dependability goals because it provide users with high-performance file access while using fewer CPU resources than standard DECnet for OpenVMS remote file access.

The DFS software supports applications that use virtual I/O and do not require write sharing. By using DFS, directory structures and files can be made available to other DFS nodes. These directory structures are given names that are registered in a DNS namespace (see Section 7.7).

Users on DFS nodes who want to access distributed directory structures specify the name of the directory structure, using the DFS MOUNT command. DFS can be used in any DECnet environment, including local area networks and wide area networks. File security is provided through the standard OpenVMS proxy mechanism. DFS also calculates end-to-end checksums to guard against network faults. ♦

7.9 Proactive Network Monitoring and Analysis Products

As a network manager or system manager, you need tools to monitor and report on error conditions so that network component faults do not interfere with requirements for 24x365 communications. The following list summarizes some of the networking products and tools available from Digital. Your Digital account representative can supply software product descriptions (SPDs) for these products. See also the network management solutions chapter in the *Networks Buyer's Guide* for detailed descriptions of the products.

- LAVC$FAILURE_ANALYSIS, provided by the OpenVMS operating system in your SYS$EXAMPLES directory. This Macro program isolates and identifies failed components in local area VAXclusters. See Section 7.3 for summary information and *VMScluster Systems for OpenVMS* for details.
- DECalert consolidates and distributes system and user alarms to appropriate personnel through a variety of alerting technologies such as pagers, electronic mail, telephone calls using DECtalk™, and graphics on workstations. As part of a high availability platform, DECalert manages, tracks, and escalates all alert notifications and support staff acknowledgements. DECAlert includes network sensors called server node watch, LAT node watch, and DECnet node watch. To form an early detection system, DECalert works with a variety of other Digital products, including VAXsimPLUS™, VMScluster Console System (VCS), NMCC/DECnet Monitor, LAN Traffic Monitor, NMCC /VAX ETHERnim, and DECtalk. You can use DECalert to create customized sensor modules for networking and other system management tasks.

Note

See Section 11.6 for information about how the data center staff of a Digital Customer Support Center (CSC) used DECalert and other products listed in this section to support CSC business requirements for 100% application availability.

- Ethernet Network Integrity Monitor (ETHERnim) software is a network management product that aids in fault isolation and configuration management of extended LANs. ETHERnim is a OpenVMS layered product that monitors a network and builds a database and graphic topology map of the extended LAN.
- LAN Traffic Monitor (LTM) provides information needed to maximize network performance and plan network growth. LTM is a tool used to analyze the use of an extended LAN. This product provides network use graphs that show the network manager the performance of the various LAN segments. LTM gathers traffic statistics from any device connected to the LAN and provides data based on nodes, addresses, and protocol types.
- DECnet Monitor is a software product for the observation and control of complex, corporate backbone networks. DECnet Monitor responds to English-like commands and graphically presents network conditions on a color (or monochrome) network topology map. DECnet Monitor features include the following:
  - Nodes can be instructed to send selected DECnet events to DECnet Monitor, helping (along with DECalert) to avoid major problems and minimize node down time because it enables a network manager to detect problems early.
  - Relational database capabilitites that provide for active or passive collection of statistics.
  - Visual displays of network configuration and performance information that simplify interpretation of data and highlight conditions requiring attention. The network topology is displayed as a logical map for recognition of network elements and potential trouble areas.
  - Histograms and bar charts that display historical information to help data center staff analyze trends and plan for growth and change.
  - Real-time event logging with predetermined polling parameters that enable data center staff to detect problems early.
- The VMScluster Console System (VCS) software product consolidates network and system management functions by replacing multiple hardcopy console terminals with a single VCS system. When VCS is linked to each device, all console messages are sent to a single console. VCS time-stamps all data received from each node and records it in a central disk file. This information can be retrieved at specific time intervals for display or printout by any terminal connected to the VCS.

  Using VCS reduces space constraints and cuts overhead. Without VCS, system managers and operators typically look for error messages (on reams of paper near each node) about conditions that might adversely affect system or network performance and reliability.
- DEC Extended LAN Management Software (DECelms) lets you, from a host VAX node, configure, manage, and monitor any LAN bridge and FDDI wiring concentrator in the extended LAN and FDDI network environment. DECelms resides on a VAX host. Corresponding management firmware resides in

the LAN bridges and FDDI DECconcentrators. The management protocol is used to communicate between the VAX host and the target LAN bridges and FDDI DECconcentrators. DECelms provides additional functionality to support the control and observation of the target LAN bridges and FDDI DECconcentrators. Features include the ability to:

- Display and modify devices, lines, physical ports, and forwarding database characteristics
- Enable and disable lines and physical ports
- Build automatically a host registry of all reachable FDDI DECconcentrators and LAN bridges within the extended LAN and FDDI environment
- Poll (automatically or manually) the devices stored in the host registry for faults, errors, and changed information
- Display errors to the alarms window of the user's display while more detailed information is being recorded in the error log file
- Process the log file keying off time or type of faults
- Display data-link counters, status, and characteristics
- Modify parameters such as operational state, forwarding database, and spanning tree characteristics

• DEC Management Control Center (DECmcc™) is a multivendor network management platform that coordinates interactions among various network modules. Using DECmcc, several network systems (including those from computer vendors other than Digital) can be monitored simultaneously. Other products, including DECalert, interface with the Digital Enterprise Management Architecture (EMA) compliant version of DECmcc via the DECmcc ALARMS functional module. (See Chapter 11 for details about how DECalert is used at the Digital Customer Support Center in Colorado.)

  DECmcc is a package of network management products. DECmcc Site Management Station provides configuration, fault, and performance management for multivendor LANs; it also provides centralized server and bridge management. DECmcc Site Management Station is based on four network management products: NMCC/VAX ETHERnim, LAN Traffic Monitor (LTM), Terminal Server Manager (TSM), and Remote Bridge Management Software (RBMS). The DECmcc Enterprise Management Station is a network management product for WANs. It uses the NMCC/DECnet Monitor product and creates a multifunction graphic display of the network topology.

  See the DECmcc documentation for details.

8 Building Dependable Software Applications

This chapter describes the software component of dependable computing systems. Clearly, building dependable computing systems is not simply a matter of using redundant hardware components. Software product or application compatibility with hardware components and other software products or applications is essential to dependability. Software incompatibility with a specific revision of hardware, the operating system, preexisting applications, or databases is a significant cause of down time and intermittent failures.

Topics covered in this chapter are:

- Using DEC Reliable Transaction Router™ for software application fault tolerance. See Section 8.1.
- Understanding and using OpenVMS features that support dependable computing systems. These features include OpenVMS software support for:
  - VMSclusters
  - Volume Shadowing for OpenVMS
  - Disk striping and data parity
  - DECamds™, an availability manager for distributed systems
  - DECdtm™, a component of OpenVMS that provides a two-phase commit protocol

  See Section 8.2 for details.
- Writing predictable, dependable code. See Section 8.3.
- Prototyping applications to build in dependability. See Section 8.4.
- Testing applications to verify their dependability. See Section 8.5.
- Understanding and using system software products that support application dependability goals, including DEC Rdb, DECforms, and VAX ACMS. See Section 8.6.
- Understanding and using Digital COHESION environment software products that manage shared information. These products include CDD/Repository for the control of data repositories that are shared by many application development organizations. See Section 8.7.

8.1 Using DEC Reliable Transaction Router for Software Fault Tolerance

People who design critical client/server applications, which often demand high availability and operate in wide area networks (WANs), face a tough challenge. How can these applications remain reliable when there are many potential failure points in the overall system? Most failures do not occur at the client or server nodes; most failures occur *between* the client and server nodes.

DEC Reliable Transaction Router (RTR) is a software fault tolerant platform used for the implementation of reliable and distributed systems. RTR uses a three-layer software model that guarantees the delivery of a transaction message from clients to servers across a network. The network can be a local area network (LAN) or a wide area network (WAN). Clients send messages to routers, which route the message to the appropriate server for execution. Reliability is provided by software fault tolerance through redundant paths and multiple systems if necessary.

The scalable three-layer model enables flexible growth of the client/server system in response to changing business requirements, with no application code changes. Applications can be developed with common 3GLs, including C++, or DECADMIRE for rapid application development. Client and server support is provided for OpenVMS AXP, OpenVMS VAX, and DEC OSF/1, with client-only support for Microsoft DOS, Microsoft Windows, and ULTRIX.

RTR is suitable for:

- Applications with multiple geographic locations
- The need to integrate distributed applications
- Requirements for high levels of reliability, availability, fault tolerance, and high-performance throughput

RTR allows applications to be written in a simple manner, and in terms of the business objectives the applications must achieve.

8.1.1 Requester and Server Processes

The main function of any transaction processing (TP) system is to provide a link between the users of the system and a database. A typical example is an airline reservation system. The travel agent (that is, the user) makes airplane seat and hotel room reservations. The database is queried to find seat and room availability, and then reservations are recorded in the database.

An application has to perform the following major tasks:

- Managing the internal data store (a database)
- Interfacing with the outside world (the end users)

This characteristic suggests a division of application function in TP systems into:

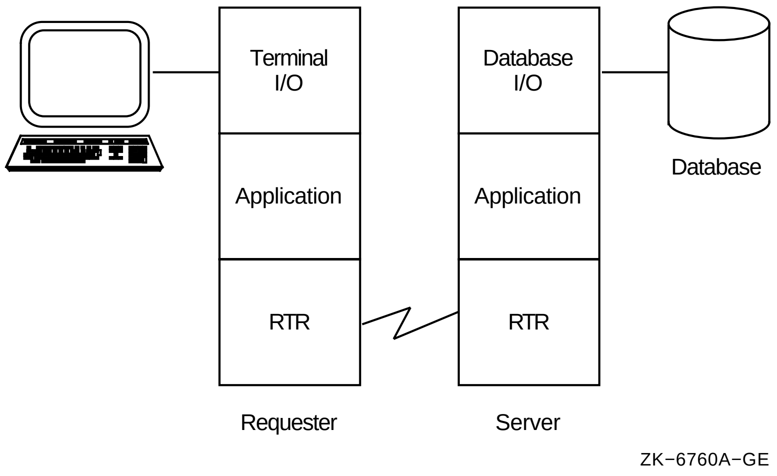
- Requester applications to interface with the end users
- Server applications to manage the data

Figure 8–1 shows how RTR allows you to split your application into requesters and servers and distribute them around the network, without burdening your applications with extra code to:

- Decide where to send messages

- Cope with network or node failures

Figure 8–1 Splitting an Application into Requesters and Servers with RTR

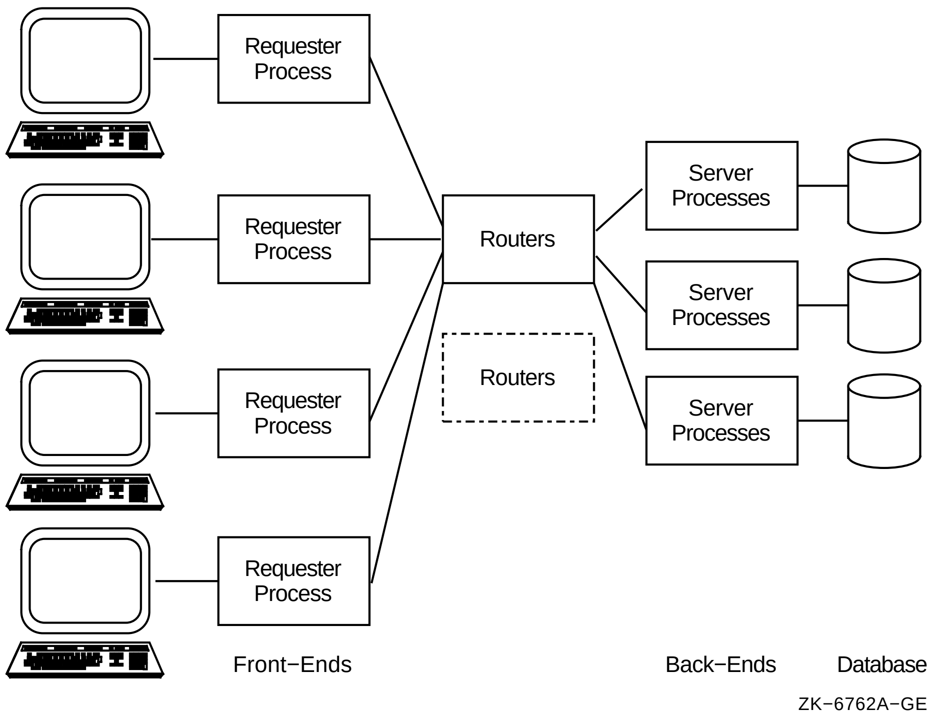


This feature lets you place elements of the application where they work best.

8.1.2 Three Layer Model

RTR is based on a three layer model consisting of front-end nodes, back-end nodes and router nodes as shown in Figure 8–2.

Figure 8–2 Three Layer RTR Model



The requester processes run on the front-end nodes. This layer allows computing power to be provided locally at the end-user site for transaction acquisition and presentation.

The server processes run on the back-end nodes. This layer:

- Allows the database to be distributed geographically
- Permits fault tolerant systems to be constructed which can cope with either node or site failures
- Allows sufficient computer resources to be deployed to meet performance requirements
- Allows performance or geographic expansion while protecting the investments made in existing hardware and application software

The router layer normally does not contain application software. This layer reduces the number of logical network links required on front-end and back-end nodes. It also decouples the back-end layer from the front-end layer so that configuration changes in the (often rapidly changing) user environment have little influence on the transaction processing and database (back end) environment.

Although RTR is based on the three layer model shown in Figure 8–2, it is not necessary for the three layers to be physically present. For a system with few front ends, the router and front-end layers can, for example, be combined. During application development and testing, all three layers can even be combined in one node. The physical locations of the front-end, router, and back-end nodes are specified using RTR configuration commands. RTR application code is completely location and configuration independent.

8.1.3 Partitioned Data Model

A key element of designing for high transaction throughput is reducing the time that users must wait for shared resources.

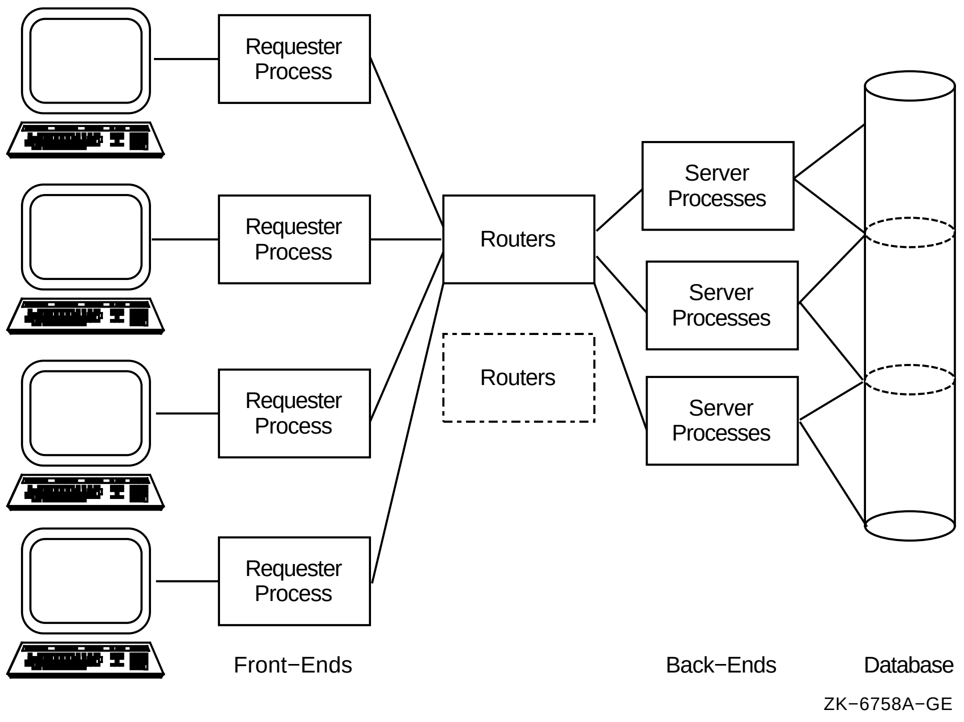
Many of the elements of a TP system can be duplicated; however, one resource that must be shared is the database. A shared database makes users compete in three ways:

- For the use of the disk
- For locks on database records
- for the CPU resources needed to access the database

These problems can be alleviated by spreading the database across a number of back-end nodes, each node being responsible for a range of values for a particular key.

RTR is designed to allow you to implement this kind of partitioned data model, as shown in Figure 8–3.

Figure 8–3 Partitioned Data Model



In this example, the database is split into three partitions, on the basis of some key.

8.1.4 Transaction Integrity

By using RTR, the design and coding of distributed applications is greatly simplified. This is because database actions can be bundled together into transactions. A requester indicates the start of a transaction by calling the $START_TX system service and indicates the end of the transaction by calling the $COMMIT_TX system service.

This scheme ensures that:

- If any hardware or software component fails while a transaction is active, the requester application does not need to cancel the effect of RTR system services already issued for the current transaction.
- If any hardware or software component fails after the requester has committed a transaction, the database will be updated during a subsequent restart. Transactions that were committed are stored and automatically replayed by RTR.

The $ABORT_TX system service allows the application to invoke the clean-up capabilities of RTR and thus avoid the need to cope with complex clean-up within an application program.

Many transactions can be initiated by the same or different applications within an RTR network: all transactions are independently handled.

8.1.5 Broadcasts

The previous RTR sections discussed how requesters and servers communicate to execute transactions. In addition, there is often a need to send unsolicited messages from a server to some or all of the requesters in the system.

An example is a commodity trading system, where the requesters are submitting orders to trade and also need to be kept informed of the latest price changes.

The RTR broadcast feature provides this capability. Requesters tell RTR which classes of broadcasts they want to receive. When a server broadcasts a message in a certain class, all the interested requesters receive the message.

It also is possible for requesters to broadcast messages to some or all of the servers using the broadcast mechanism.

8.1.6 RTR Dependability Features

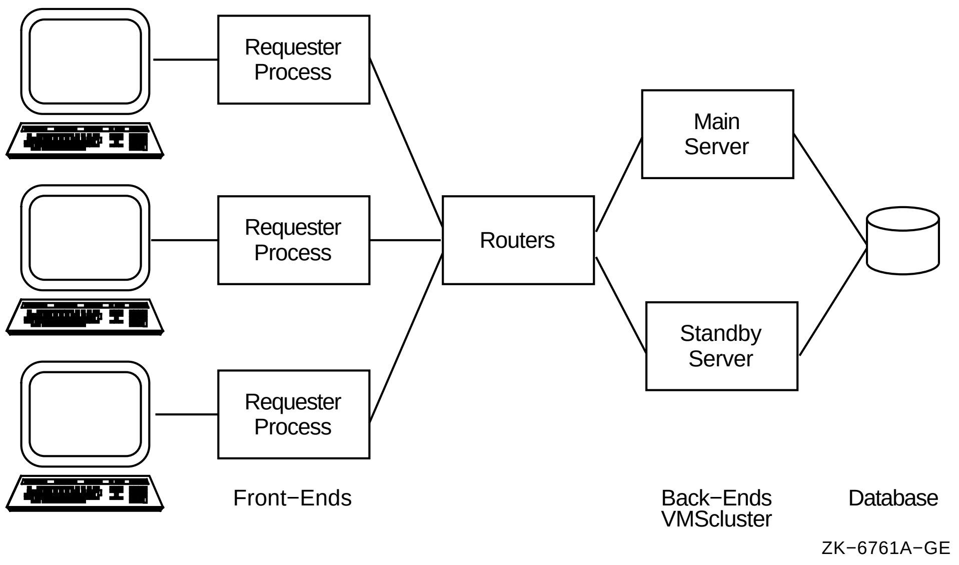
This section describes the dependability features provided by RTR.

8.1.6.1 Standby Servers

Standby servers are *spare* servers that automatically take over from the main back-end if it fails. This takeover is transparent to the application.

Figure 8–4 shows a simple standby configuration. The two back-end nodes are members of a VMScluster and are both able to access the database.

Figure 8–4 Simple Standby Configuration

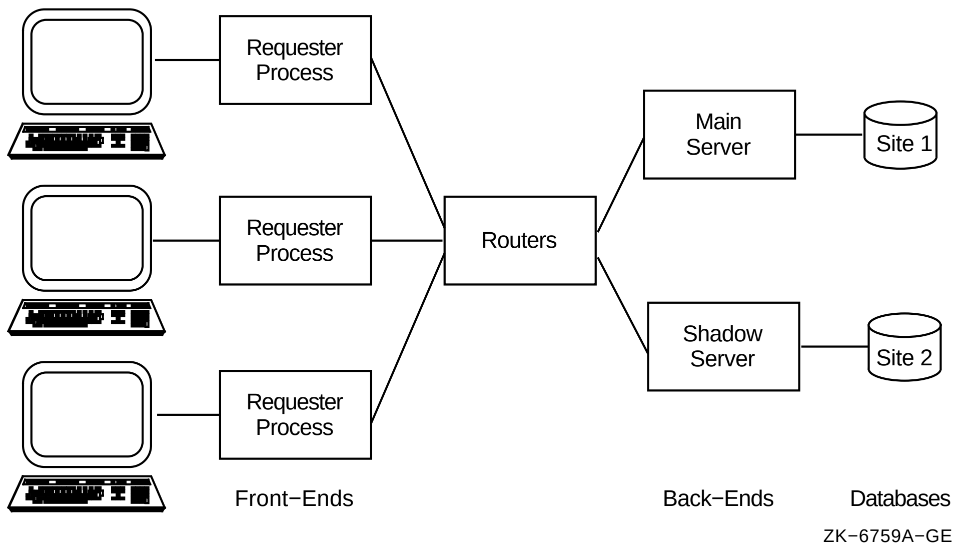


For any one key-range, the main server runs on one node while the standby server runs on the other node. The standby server process is active, but RTR does not pass any transactions to it. Should the first node fail, RTR starts passing transactions to the standby server. Note that it is possible for a single node to contain the primary servers for one key-range and standby servers for another key-range. This allows the nodes of a VMScluster to act as standby for other nodes without having idle hardware.

8.1.6.2 Shadow Servers

Shadow servers are servers on separate back ends that handle the same transactions in parallel on identical copies of the database. The databases are mirrors. Figure 8–5 shows a simple shadow configuration. The main server at site 1 and the shadow server at site 2 both receive every transaction for the key-range that they have requested. If site 1 fails, site 2 continues to operate and the service is uninterrupted.

Figure 8–5 Simple Shadow Configuration



8.1.6.3 Router Failover

Front-end nodes automatically find another router if the one being used fails. This reconnection is transparent to the application. Front ends may also be configured to choose routers with the least loading.

8.1.6.4 Cooperative Transaction Recovery

If the original router coordinating a transaction fails, back-end nodes select another router that can ensure correct transaction completion.

8.1.6.5 Back-End Restart Recovery

Transactions that had not been committed by the servers at the time of a back-end failure are replayed to the servers from a disk journal when the failed back end restarts.

8.1.6.6 Transaction Message Replay

Transaction messages that are lost in transit are resent when possible. The front-end and back-end nodes keep an in-memory copy of all active messages for this purpose.

8.1.6.7 Link Failure Recovery

In the event of a communications failure, RTR tries to reconnect the link (or links) until it succeeds.

8.1.7 Concurrent Servers

Concurrent servers are server processes running on the same node that handle the same key-range. This is useful for increasing throughput, especially when the database only allows one active transaction per process. The types of concurrent servers are summarized in the following list:

- Concurrent servers handle similar transactions and run in the same node.
- Shadow servers handle the same transactions and run in different nodes.
- Standby servers do not handle any transactions unless a main node fails.
- Call-out servers run on back ends or routers. They only vote, as explained in Section 8.1.11.

8.1.8 Flexibility and Growth Provided by RTR

RTR lets you cope easily with changes in:

- Network demand
- How users access the data
- The volume of data

Because an RTR-based system can be built using multiple processors in each functional layer, your system can grow incrementally and you can avoid unused capacity at each stage. With your system still up and running, you can:

- Create and delete concurrent server processes
- Add or remove nodes (front end, router or back end)
- Change the key-ranges used to route transactions to the servers

By providing these features, you may not need to make such accurate long-range capacity forecasts.

RTR provides a comprehensive set of tools that lets you monitor the volume of traffic passing through the system. You can react quickly to unexpected load-changes by making dynamic changes to the system configuration.

8.1.9 Failure Scenarios

This section describes how RTR recovers from a number of different types of hardware and software failure.

8.1.9.1 Loss of a Back-End Node

If standby or shadow servers are available on another back-end node, operation of the rest of the system will continue without interruption, using the standby or shadow servers.

If a back-end node is lost, any transactions in progress are recovered when the back end restarts, or are recovered by a standby if one is present. This means that the distributed database will be brought back to a transaction-consistent state.

8.1.9.2 Loss of a Router

If a router fails and any other router node is available, all in-progress transactions are transparently rerouted via another router. System operation will continue without interruption using the available routers.

If no other router nodes are available, the back ends wait till any router node recovers, and then complete all in-progress transactions.

8.1.9.3 Loss of a Communications Link

If a communications link is lost, then:

- RTR continually tries to establish the failed link.
- Database recovery is performed when the link is established again.

8.1.9.4 Loss of a Front-End Node

If a front-end node is lost, then:

- All transactions committed but not completed on the front-end node at the time of failure will be completed.
- All transactions started but not committed on the front-end node at the time of failure will be aborted.
- Failure to deliver a transaction completion message to the requester on the front end causes RTR to write an error-log entry on a back end or router node.

8.1.10 RTR Facilities

Many applications can use RTR at the same time without interfering with one another.

This is achieved by defining a separate facility for each application. A facility can be thought of as being an application's own named invocation of RTR.

When an application calls the $DCL_TX_PRC system service to declare itself as a requester or server, it specifies the name of the facility it wants to use. The use of facilities allows the creation of massively parallel networked systems supporting a number of independent applications.

8.1.11 Call-Out Servers

RTR has a **call-out server** capability that allows security checks to be carried out on all requests in a given facility. This is particularly appropriate in systems where the organization providing the back-end nodes wants to protect itself against potentially hostile front-end nodes not under its direct control.

Call-out servers can run on back-end or router nodes. The call-out servers receive a copy of every transaction that is delivered to or passing through the node. Because call-out servers are also asked to vote on the outcome of the transactions they receive, they can effectively veto any transaction that does not pass the security checks.

Such security checks can also be included in the conventional database update servers but call-out servers offer the following advantages:

- The security check can run in parallel to the database updates thus improving response times
- The security check can run on the router hardware if there is a physically separate routing layer

- The security checking code is completely separated from other application code

Because this technique relies on backing out unauthorized transactions, it is less suitable when a large proportion of the transactions are expected to fail the security check.

8.1.12 Application Programming Interface and Documentation Pointers

The RTR application programming interface is available in two forms:

- System services that are described in *DEC Reliable Transaction Router for OpenVMS Application Programmer's Reference Manual.*
- A DCL interface that lets you write simple RTR applications for the purposes of configuration testing. It is described in the *DEC Reliable Transaction Router for OpenVMS System Manager's Manual.*

8.2 OpenVMS Dependability Features

The OpenVMS dependability features include:

- Operating system support of redundant functional units. See Section 8.2.1.
- Support of VMScluster application environment topologies. See Section 8.2.2.
- The Digital Availablility Manager for Distributed Systems (DECamds). See Section 8.2.3.
- Volume Shadowing for OpenVMS. See Section 8.2.4.
- Disk striping and parity. See Section 8.2.5.
- Digital Distributed Transaction Manager (DECdtm), an OpenVMS component that provides support for the two-phase commit protocol. The protocol ensures the atomic nature of a distributed transaction that spans multiple files, databases, or both. See Section 8.2.6.
- DEC File Optimizer for OpenVMS, a disk file defragmentation product, and an OpenVMS subfunction called **movefile** that lets programmers create their own disk defragmentation applications. Section 8.2.7 introduces DEC File Optimizer for OpenVMS and **movefile** . Section 9.6.2 provides more details.
- RMS journaling. See Section 8.2.8.
- An OpenVMS batch and print queuing system that provides, in a VMScluster environment, failover of batch and print jobs that were active at the time the local node failed. Section 8.2.9 introduces the clusterwide batch and print queuing and failover feature, and Section 9.7 provides more details.

8.2.1 OpenVMS Support of Redundant Functional Units

OpenVMS controls the hardware in most AXP and VAX configurations. As discussed in Chapter 1, given that hardware components will eventually fail, the way to provide dependable configurations is by using appropriate levels of redundancy. The following list outlines the various methods that OpenVMS uses to recover from hardware failures and to provide dependable service to your users:

- Central processing units (CPUs)

When a system kernel is configured with more than one CPU (that is, a symmetrical multiprocessing (SMP) configuration), OpenVMS first attempts to recover in real time. Certain types of CPU failures at specific times can be tolerated by OpenVMS without crashing. Then, only the currently active process on the failed CPU is lost. Usually, however, a failing CPU (unless part of a VAXft system) causes OpenVMS to crash. An immediate reboot utilizing the surviving CPU(s) would then be possible with an SMP system. Figure 6–2 depicts an SMP system kernel with two CPUs.

- Main memory subsystems

  Many AXP and VAX memory subsystems use an error correcting code (ECC) technology to transparently correct single-bit errors (and detect double-bit errors) that might occur. The least expensive memory subsystems typically use parity error detecting schemes that notify OpenVMS of a problem. In either case, if an uncorrectable error occurs on a memory page, OpenVMS attempts to isolate the problem by placing the page on a *bad page* list. Doing so prevents its reuse (the process using the page must be deleted). If the bad page happens to be one assigned to a critical part of the OpenVMS executive, the error will cause OpenVMS to crash immediately. This *fail-stop* behavior protects the integrity of your data by preventing a flawed OpenVMS executive from proceeding. OpenVMS also performs a memory subsystem test each time it initializes itself.

- Computer interconnect I/O adapters

  If you install more than one CI adapter in a system kernel, OpenVMS will spread the I/O load over the total set of adapters. Further, if one of the adapters should fail, the transfers are reexecuted on the surviving adapter(s). All of the hardware failures described in this section are recorded in the OpenVMS error log file for later analysis.

- Network I/O adapters

  Many AXP and VAX configurations use their Ethernet adapters as their only network adapter. In the future, a growing percentage of AXP and VAX systems will use multiple Ethernet and Fiber Distributed Data Interface (FDDI) adapters for network (or VMScluster) connections. The DECnet for OpenVMS software supports multiple adapters on a system and multiple lines between nodes, and balances the network traffic across them. In the event of a line failure, the surviving lines would continue to carry network I/O traffic without loss of data. Multiple synchronous lines would operate in the same redundant fashion.

  Refer to Section 7.2 for information about local area VMScluster support for multiple LAN adapters to Ethernet and FDDI cable.

- Disk controller modules

  In AXP and VAX system kernels that allow multiple controllers, OpenVMS supports multiple paths to dual-ported disk drives. In this way, if a controller module that is connected to port A of a key disk fails, OpenVMS can continue to use the controller module that is connected to port B of the disk drive. Figure 6–6 illustrates disk drives with connections to both ports. In that diagram, however, the disk drives are not connected to disk controller modules but to hierarchical storage controllers (HSCs).

- HSCs

These disk and tape controllers, by virtue of their being CI nodes, are connected to every AXP or VAX host on that CI star coupler. They are very frequently installed in redundant pairs because OpenVMS supports the automatic failover from a failed HSC to the surviving one. If each disk drive has one of its ports connected to an HSC, then all the nodes have redundant paths to those disk drives. Figure 6–6 depicts HSC units connected in this dual configuration.

Note

The I/O protocol used with HSCs and Volume Shadowing for OpenVMS controller modules ensures that no data is lost because of HSC or controller failover.

- Disk drives

  Volume Shadowing for OpenVMS is used for maintaining multiple disk drives with the same contents. Any disk writes are written to all members of the shadow set while disk reads take the most efficient path to only one member of the shadow set. Figure 6–7 depicts two independent shadow sets that each have shadow members in different cities.

  The key advantage to using volume shadowing is the ability to survive a crash or other catastrophic failure and increase data availability. Depending on the configuration, volume shadowing can keep the data available in the event of failure of the media, any part of the drive itself, cables, the controller, the network, the interconnect, or some subset of the nodes of the VMScluster. No data or processing is lost because the failover is virtually transparent to application programs (except for a slight pause in I/O). With Volume Shadowing for OpenVMS, when a replacement disk is added as a shadow set member the overhead I/O operations necessary to update this disk are transparent.

  See Section 8.2.4 for more details.

  The mass storage control protocol (MSCP) server also contributes to the data availability of OpenVMS disks. MSCP supports a wide variety of configurations and enables disk drives that are not connected to HSC units to be mounted on all nodes of a VMScluster. Two AXP or VAX nodes can be connected to the two ports of a disk drive that use Volume Shadowing for OpenVMS and can make the data on the disk available to the other VMScluster nodes in a fashion similar to dual HSC units. A disk drive could also have one port connected to an HSC while the other was connected to a node directly, assuming the controller is connected correctly.

- Terminal servers

  These servers provide the primary access to OpenVMS for most terminal users. They provide redundancy in the following ways. When first logging in to the system, a terminal user who connects to a VMScluster via a Digital terminal server would be automatically connected to the AXP or VAX node with most available resources. Once connected, the terminal user could press the break (the keyboard label) key and set up additional sessions on the same node or other nodes. Then, if the AXP or VAX node that is being used happens to fail, the terminal server attempts to fail over the terminal to another session on a surviving node.

The Digital VT300-series and VT400-series terminals have two communications jacks. The dual jacks let a single terminal physically connect to two terminal server ports simultaneously, which may or may not be on the same terminal server. This capability enables the terminal user to continue work even if one of the Ethernets, terminal servers, or Ethernet adapters had failed. Figure 6–2 depicts a terminal (with attached printer) that is connected to two different terminal servers. The terminal servers happen to be connected to two different Ethernet segments.

- VMScluster console systems

  In many of the hardware configurations in Chapter 6, a VMScluster Console System (VCS) software product could be used to provide efficient control of all the nodes in the VMScluster. The VCS need not be a single point of failure because twin VCS systems could be connected to all the nodes and set up so that the secondary VCS would automatically take over if the primary VCS failed. Figure 4–1 depicts a CI-based VMScluster with one VCS connected to the VAX consoles. HSC and other consoles (other than VAX consoles) could be connected as well.

8.2.2 VMScluster Application Environment Topologies

Almost every feature of VMScluster systems, including transparent access to common data storage with record-level locking, incremental growth while protecting current capital investments, and the ability for the system to continue providing service in the face of many kinds of failures is available to your applications without special design or coding considerations. OpenVMS applications that function correctly on single node AXP or VAX systems (fault tolerant or not) will continue to function correctly on the various types of VMScluster systems, without recoding, recompiling, or relinking.

However, applications can be optimized for their environment. Some techniques are described briefly in this section. See Chapter 10 for information about how Digital can help you optimize your applications.

8.2.2.1 Application Scaling Considerations

Your staff must exercise good design practices and algorithms when creating your application system. A program's ability to grow in scale efficiently is particularly important.

Growth in scale occurs when the number of simultaneous users of a program increases. This can happen when a more powerful system kernel is installed or when additional system kernels are added to your VMScluster system. In some cases, previously undetected performance bottlenecks in the application design may surface at the higher level of loading. An example would be the practice of storing the next free invoice number in the first record of the master file. While the disk containing such a frequently accessed record may be able to sustain the throughput to handle 20 or 30 simultaneous users, 200 or 300 users might overload the same disk with I/O requests.

One alternative to application redesign might be the installation of a faster disk, such as the Digital ESE-20 Solid State Disk, or a memory resident disk, such as the DECram for OpenVMS Device Driver (MDDRIVER) software. Because it provides non-volatile storage, the Digital ESE-20 is optimized for both read-intensive and write-intensive applications. By utilizing main memory for storage, DECram software is optimized for read-intensive or scratch file applications.

Another dimension of scaling is database size. Use care with the data file and database access practices of the application system. Sequential scanning techniques that may provide acceptable performance on files with 1000 records might cause excessive delays if attempted on files with 100,000,000 records.

8.2.2.2 Resource Contention Considerations

Any large computing system, whether a mainframe such as a DEC 10000 AXP system or a fault tolerant system such as a VAXft 3000-810, can be constrained by resource contention problems. OpenVMS uses sophisticated algorithms to allocate system memory and CPU time to multiple users (or programs) with a minimum of contention. However, the application programs themselves have a large amount of control over some common system resources, such as data files and databases.

To maintain data integrity in a multiprogramming, multiuser environment, Digital database products like DEC Rdb that use the OpenVMS distributed lock manager employ record-level locking. While record-level locking allows greater concurrency (and therefore less contention) than file-level locking, the potential for lock conflicts (contention) still exists, particularly with large numbers of users. Some programming practices are particularly prone to causing contention problems.

If a program reads several records for update (which usually causes the system to lock them) and then waits for user input before rewriting the records (which would unlock them), there is a possibility that a second user's program may attempt to read one of the locked records during that period. The second user's program would typically wait until the first user's program released the records. If the first user decided to go to lunch before finishing the transaction, the second user (and other users attempting to access those records) would continue to wait, unless the program were sophisticated enough to detect the problem and inform the user.

A better approach to preventing such record contention would be for the program to read its records and unlock them. When the user has finished updating the information, the application should read the records again to see if they have changed in the interim, and take appropriate action if warranted.

The following sections outline methods for addressing contention, scaling, and synchronization issues.

8.2.2.3 Independent Processes Paradigm

A common application system paradigm in a multiprogramming, multiuser operating system like OpenVMS is that of independent user processes. In this scenario, each application program is designed as if it had the entire computing system to itself. The operating system's job is to mask the fact that other users are, in fact, using the system as well. Common examples of this techniques include:

- Overlap of one user's I/O with another user's processing
- Separate process address spaces
- Automatic record locking and unlocking

The individual invocations of each such application program usually share only their common data files and databases. Because data storage is normally transparently accessible from all nodes in a VMScluster system, programs can execute on any node in the same fashion as they would on an AXP or VAX system that was not part of a VMScluster.

Note

In some cases, an application program may have a dependency on certain nodes in a VMScluster system. Those programs that expect the presence of a vector processor would perform adequately only on those system kernels in the VMScluster with such hardware.

Some programs may take advantage of operating system features that are present in one version of OpenVMS but not present in an earlier version. Those programs may need to be restricted to certain nodes if the VMScluster is operating with multiple versions of OpenVMS.

Applications written with this paradigm have the advantage of not needing to know whether they are executing in a VMScluster environment. However, they still require careful analysis and design to prevent the previously mentioned contention and scaling problems.

Another implication of this application system paradigm is the loss of some production work if one of the VMScluster nodes fails. While the Digital file and database subsystems are engineered to protect data integrity in the face of such a failure, the user processes that were executing on the failing node are lost. Batch jobs can be written to restart at defined points in the middle to avoid redoing already completed processing steps, but interactive users must log in again to the VMScluster. Digital terminal servers can alleviate this problem by allowing users to initially log in to more than one VMScluster node and then by automatically switching the user's terminal to a surviving node when necessary.

See the *OpenVMS System Manager's Manual* for more information on restarting batch jobs. See the *Networks Buyers Guide* for more information on Digital terminal servers.

8.2.2.4 Distributed Arbitration Paradigm

Some application systems may require an invocation of a program to be aware of the operation of other invocations of the same program. Designs that bypass the automatic record locking of OpenVMS would need to control their own access to shared resources.

A noncomputer example of distributed arbitration would be automobiles queuing up at an intersection with stop signs for all directions. By convention, the drivers wait their turn to cross the intersection. Clearly, this protocol works only if all drivers share a common understanding of the dynamic state of the intersection. If all the drivers do not watch carefully to note the order in which automobiles arrive at the intersection, there could be two drivers who simultaneously concluded that it was their turn to proceed.

This paradigm has the advantage of not relying on a sole arbiter (such as a traffic officer) that would be a single point of failure. However, a high level of sophistication is required to implement such a paradigm with high levels of efficiency.

8.2.2.5 Client/Server Computing Paradigm

In the client/server environment, the applications are client programs that make interprocess calls to one or more dedicated server processes that provide key functions such as database access or numerical computations. The success of this approach depends on highly efficient interprocess communications mechanisms and robust process control primatives in the operating system. For more information on how Digital provides these features, see the Network Application Support (NAS) documentation from Digital. Start off by reading the *NAS Overview*.

One major advantage of this paradigm is the reduction in contention. If a database is accessed by five server processes (each serving 20 clients), the locking contention is much less than if the same database was being accessed simultaneously by all 100 programs. Performance benchmarks have shown that the aggregate transaction throughput is higher in the former scenario. Additional data access efficiencies are gained because a single server process can make more effective use of large buffer caches and use techniques like bundling several update operations into larger but fewer write I/Os.

Compute servers are another potential use of this paradigm. If your VMScluster system has only one node with vector processors installed (such as a VAX 9000 Model 420VP), the best way to use that investment may be to create a few compute server processes that execute only on that node. They would provide high performance computations (such as matrix inversions) for the entire VMScluster and the portions of the client programs that cannot use vectors would continue to execute on their local node. See the VAX FORTRAN™ documentation for information about the high performance option of the VAX FORTRAN compiler.

A disadvantage of the client/server approach is the level of sophistication required to engineer multiuser, multithreaded servers with high levels of efficiency and robustness. In particular, to prevent servers from becoming single points of failure in a VMScluster system, provisions must be made for automatically and quickly recreating those processes (in recovery mode) on surviving nodes, when necessary. The synchronization techniques in the following section can be useful for these situations.

Besides offering Network Application Services (NAS) to facilitate the creation of client server applications, Digital also supports a very robust client server paradigm with its online transaction processing environment, DECtp. Most of the complexity of the client server environment is handled transparently by the environment, letting your programming staff concentrate on solving the business problem at hand.

The creation and management of multiuser server processes are handled by the environment, as is the automatic failover to surviving VMScluster nodes in the event of failures. Transaction queuing is used to ensure no loss of committed transactions and the environment is fully integrated with Digital file and database subsystems. Like NAS, within the DECtp environment, robust applications are not limited to the boundaries of a VMScluster, but may be distributed across the local area or wide area network. See the *VAX ACMS Guide to Creating Transaction Processing Applications* and the *DECtp Desktop for ACMS Progamming Guide* for information about ACMS and Desktop ACMS.

8.2.2.6 Synchronization Techniques

If your application system is to gain optimal advantage of the VMScluster environment, your programs may need to access information about the current state of the VMScluster at any time. The system information services, SYS$GETSYI, F$GETSYI, and LIB$GETSYI provide a wide variety of system and VMScluster state information. For instance, one item of information is the CLUSTER_MEMBER parameter, which is either true or false, depending on the node's current status.

See the *OpenVMS Programming Concepts Manual* and the *OpenVMS System Services Reference Manual* for more information on calling the SYS$GETSYI system service. See the *OpenVMS DCL Dictionary* for more information on calling the F$GETSYI lexical function in DCL. See the *OpenVMS RTL Library (LIB$) Manual* for more information on calling the LIB$GETSYI run-time library routine.

If the distributed parts of your application do not need to synchronize with one another very often, it may be feasible to pass synchronization control and data through a shared disk file. Data storage is commonly accessible from all nodes in the VMScluster. However, it can suffer from the same frequently accessed record performance bottleneck as seen in Section 8.2.2.1.

A much more efficient method of passing synchronization control and data between VMScluster members is the OpenVMS distributed lock manager. This subsystem is used by many other parts of the OpenVMS operating system that require serialization of access to common resources (such as files), or notification of events (such as another program attempting to lock a resource that is currently locked), and sometimes the fast delivery of small amounts of data between nodes.

Because of the event notification mechanisms built into the OpenVMS distributed lock manager, application systems can use the client server topology with the security of quickly knowing when a VMScluster node has failed. Prompt notification is necessary if server restart and recovery operations are to be performed on a surviving node with a minimum of application down time. See the *OpenVMS System Manager's Manual* for more information on the OpenVMS distributed lock manager.

8.2.3 Availability Manager for Distributed Systems

Digital's Availability Manager for Distributed Systems (DECamds) is a real-time monitoring, diagnostic, and correction tool used by system managers to improve availability. DECamds collects and analyzes data from multiple nodes simultaneously, directing all output to a centralized DECwindows display. The analysis detects resource availability problems and suggests corrective actions.

8.2.3.1 Advantages of Using DECamds

DECamds helps system managers improve system availability and offers the following features:

- Alerts users to resource availability problems, suggests paths for further investigation, and recommends actions to improve availability
- Centralizes management of remote nodes within an extended local area network (LAN)
- Allows real-time intervention, including adjustment of node and process parameters, even when remote nodes are hung
- Adjusts to site-specific requirements via customization options

Anyone responsible for managing or monitoring OpenVMS systems can use DECamds. It can be used in two different modes:

- Observation mode: To continuously watch resource usage and fine-tune system availability.
- Reaction mode: For a specific problem, to quickly identify the problem and, when possible, to fix it.

8.2.3.2 DECamds Functions

DECamds has two parts:

- The driver

  The driver runs on remote nodes to gather data and implement fixes. The driver is a pseudo device driver image, running at a high interrupt priority level (IPL), so it functions even when a remote node is hung.

  Note

  Collecting data at an elevated IPL, however, prevents DECamds from collecting nonmemory resident data, restricting some data collection in process space.

  You install, load, and start the driver on all nodes you will monitor.

- The console

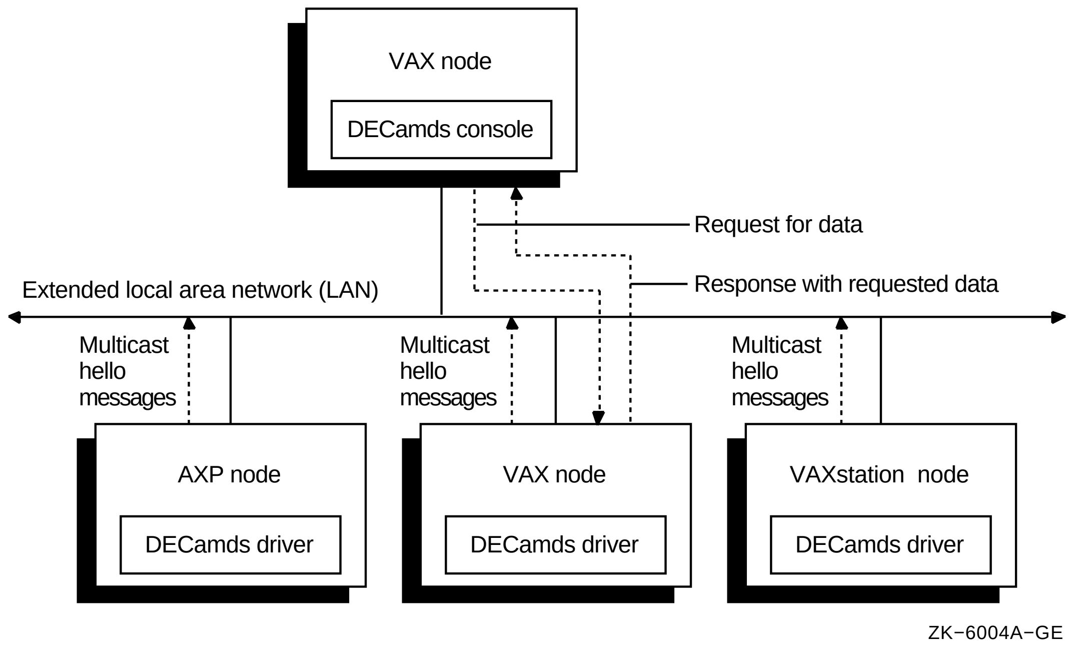
  The console runs on the local node and directs one or more remote drivers to gather system and process data and return it to the console. The console analyzes the data and displays the results on a DECwindows device. The console also directs the driver to implement fixes at your request.

  After you install and start the console and driver, you use the console to observe and troubleshoot availability problems.

The DECamds console and driver communicate over a private Ethernet protocol.

Figure 8–6 illustrates the model used by DECamds.

Figure 8–6 DECamds Model



The console displays two primary windows to help you navigate through the data:

- The Event Log window—Lists resource availability problems, called **events**. For each event, you can follow a path to investigate the underlying problem. For certain events you can also fix the problem.
- The System Overview window—Graphically displays summary data about CPU, memory, and process I/O usage for the systems DECamds monitors. This window lets you view data and find problem areas.

Using either the Event Log or System Overview window, you can:

- Collect additional data at the Group, Node, or Process level.
- Display data and refine the analysis by changing how data is sorted and filtered.
- Implement fixes to correct resource availability problems.

Using the DECamds console, you can direct the DECamds driver to implement corrective actions to remedy problems such as a runaway process or lock contention without rebooting. In extreme circumstances, you can implement a fix to intentionally crash a node.

8.2.4 Volume Shadowing for OpenVMS

A key component of overall computing system dependability is availability or accessibility of data. Volume Shadowing for OpenVMS provides high levels of data availability by allowing shadow sets to be configured on a single-node system or on a VMScluster system, so that continued access to data is possible despite failures in the disk media, disk drive, or disk controller. For shadow sets whose members are local to different VMScluster nodes, if one node serving a shadow set member shuts down, the data is still accessible through an alternate node.

Although you can create a shadow set that consists of only one disk, you must mount two or more volumes in order to shadow or to maintain multiple copies of the same data. This configuration prevents failure of a single disk drive or deterioration of a single volume.

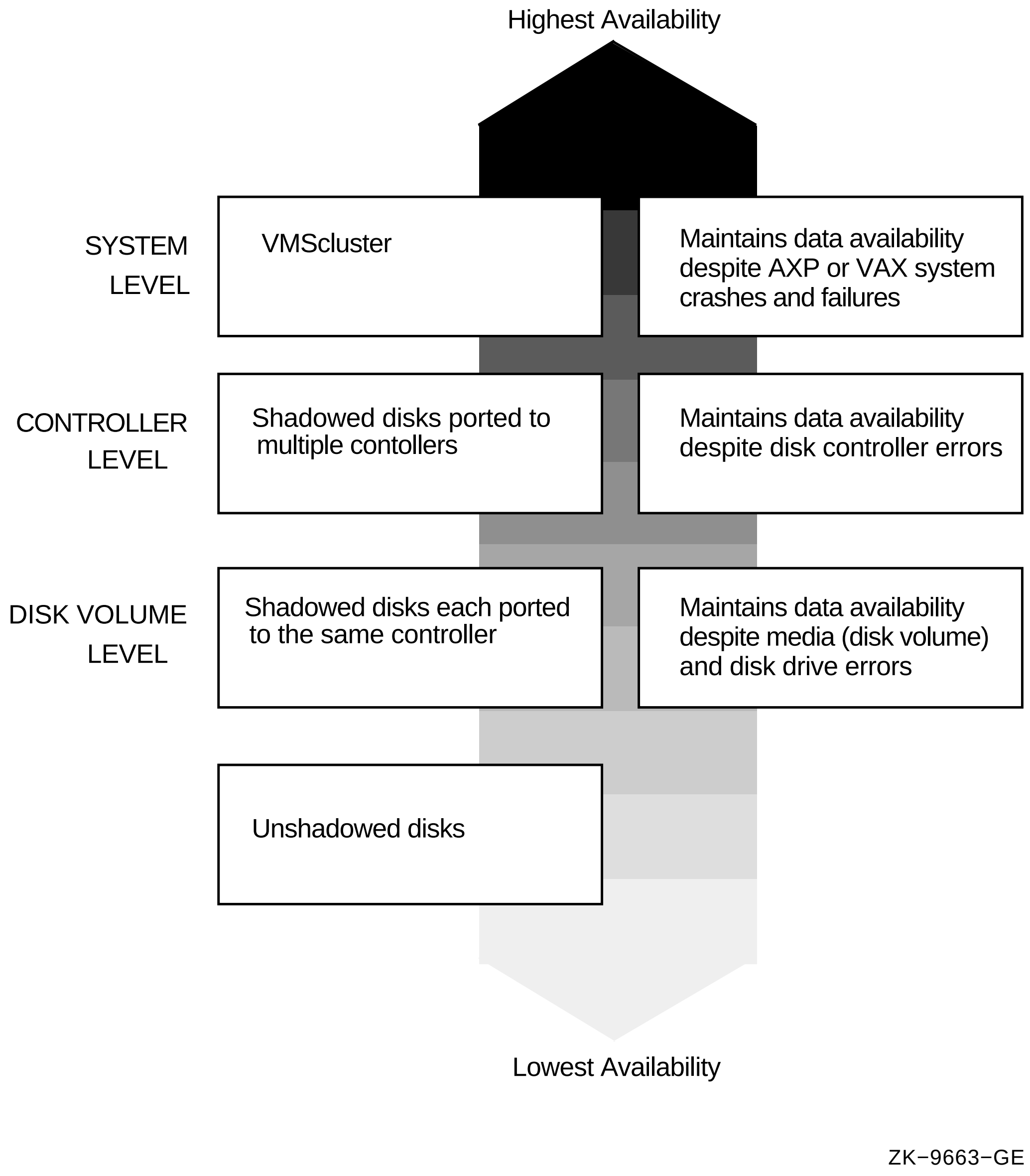
For example, if one member fails out of a shadow set, the remaining member can be used as a **source** disk whose data can be accessed by applications at the same time the data is being copied to a newly mounted **target** disk. Once the data is copied, both disks contain identical information and the target disk becomes a complete source member of the shadow set.

Using two controllers provides a further guarantee of data availability in the event of a single-controller failure. When setting up a system with volume shadowing, you should connect each disk drive to a different controller I/O channel whenever possible. Separate connections help protect against either failure of a single controller or of the communication path used to access it.

Using a VMScluster system (as opposed to a single-node environment) and multiple controllers provides the greatest data availability. Shadow sets can comprise either member units on different controllers or MSCP servers.

Figure 8–7 provides a qualitative, high-level classification of how you can achieve low to high levels of physical data availability in different types of configurations.

Figure 8–7 Levels of Availability


Highest Availability
SYSTEM LEVEL
VMScluster
Maintains data availability despite AXP or VAX system crashes and failures
CONTROLLER LEVEL
Shadowed disks ported to multiple contollers
Maintains data availability despite disk controller errors
DISK VOLUME LEVEL
Shadowed disks each ported to the same controller
Maintains data availability despite media (disk volume) and disk drive errors
Unshadowed disks
Lowest Availability
ZK−9663−GE

Section 8.2.4.1 describes how you can configure your shadowed system to achieve high data availability despite physical failures.

8.2.4.1 Repair and Recovery from Failures

A common failure that makes data unavailable is a communication failure. Communication errors fall into the categories shown in Table 8–1. A host node can detect communication failures any time data is transferred between the host computer and a controller. Table 8–1 describes the types of failures and the actions the volume shadowing software takes to repair or recover from the error.

Table 8–1 Types of Device Failures

Type	Description
Controller error	Results from a failure in the controller. If the failure is recoverable, processing continues and data availability is not impacted. If the failure is nonrecoverable, shadow set members connected to the controller are removed from the shadow set, and processing continues with the remaining members. In configurations where disks are dual-pathed between two controllers, and one controller fails, the shadow set members fail over to the remaining controller and processing continues.
Unit or drive error	Signifies that the mechanics or electronics in the device failed. If the failure is recoverable, processing continues. If the failure is nonrecoverable, the node that detects the error removes the device or unit from the shadow set.
Data errors	Results when a device detects corrupt data. Data errors usually result from media defects that do not cause the device to be removed from a shadow set. Depending on the severity of the data error (or the degree of media deterioration), the controller: • Corrects the error and continues • Corrects the data and revectors it to a new logical block number (LBN) In situations where read data is not correctable by the controller, volume shadowing replaces the lost data by retrieving it from another shadow set member and writing the data to the revectored LBN on the original shadow set member.

When a failure occurs, the first node to detect the failure must decide how to recover from a failure in a manner least likely to affect the availability or consistency of the data. The node that discovers the failure determines its course of action as follows:

- If no members of a shadow set can be accessed by the node, that node does not attempt to make any adjustments to the shadow set membership. Rather, it assumes that another node, which does have access to the shadow set, will make appropriate corrections.
- Provided that at least one member of the shadow set is accessible by the node that detected the error, that node will attempt to recover from the failure. The node repeatedly attempts to access the failed shadow set member within the period of time specified by the system parameter SHADOW_MBR_TIMEOUT. If access to the failed disk is not established within the time specified by SHADOW_MBR_TIMEOUT, the disk is removed from the shadow set.

Handling of shadow set recovery and repair differs depending on the type of failure that occurred and the hardware configuration. In general, devices that are inaccessible are failed over to other controllers whenever possible or are removed from the shadow set. Errors that occur as a result of media defects or data corruption can often be repaired by revectoring the bad block and copying good data from other shadow members.

8.2.4.2 Sample Shadow Set Configurations

This section illustrates only a few levels of data availability obtainable through Volume Shadowing for OpenVMS. The sample systems, while intended to be representative, are hypothetical; they should be used only for general observations about availability and not as a suggestion for any specific configurations or products.

Refer to *Volume Shadowing for OpenVMS* for details.

Figure 8–8 presents a VMScluster system with two AXP processors connected to multiple RF disks on a DSSI interconnect. The DSA1 and DSA2 virtual units represent the two shadow sets and are accessible through either processor. This configuration offers both an availability and a performance advantage. The shadowed disks in this configuration are highly available because the satellite nodes have access through either of the DEC 4000 systems. Thus, if one DEC 4000 member fails, the satellites can access the shadowed disks through the remaining DEC 4000 system. In addition, this configuration offers a performance advantage by utilizing an interconnect separate from the Ethernet for I/O traffic. In general, you can expect better I/O throughput from this type of configuration than from an Ethernet-only VMScluster system.

Figure 8–8 Configuration of a Shadow Set (Highly Available Local Area VMScluster)

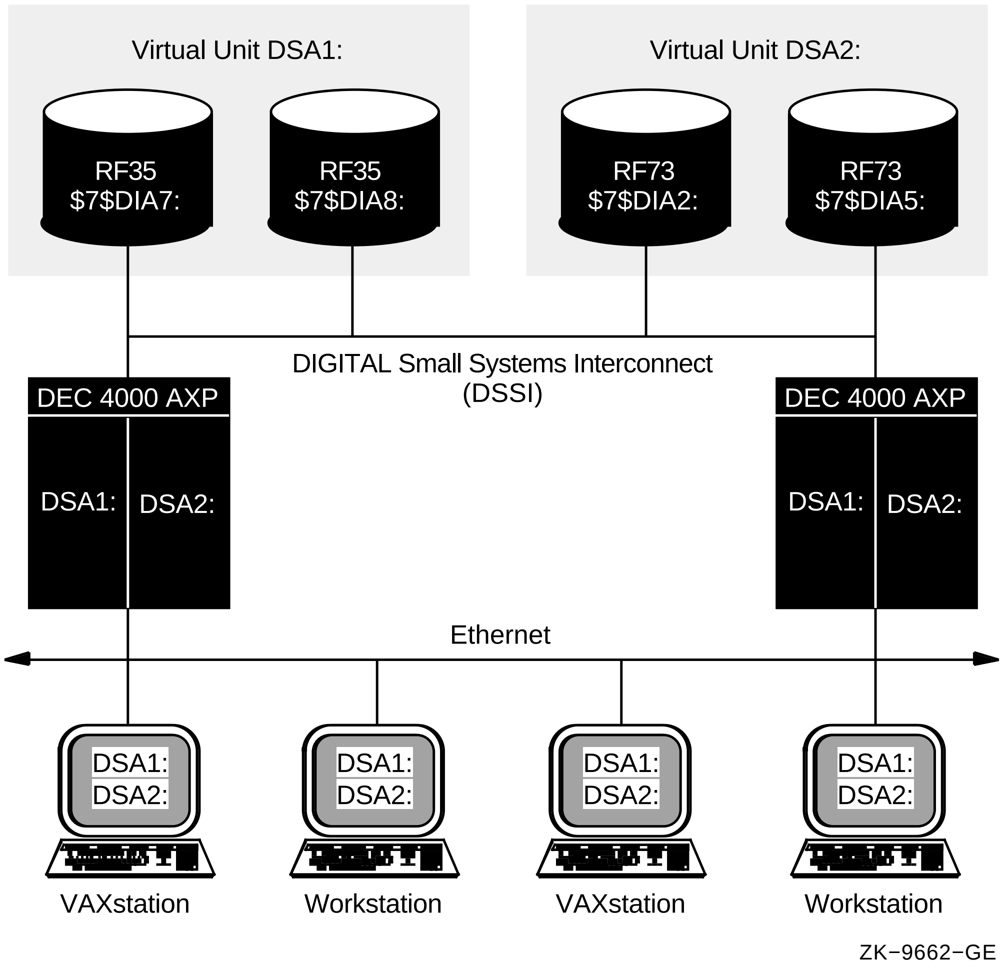
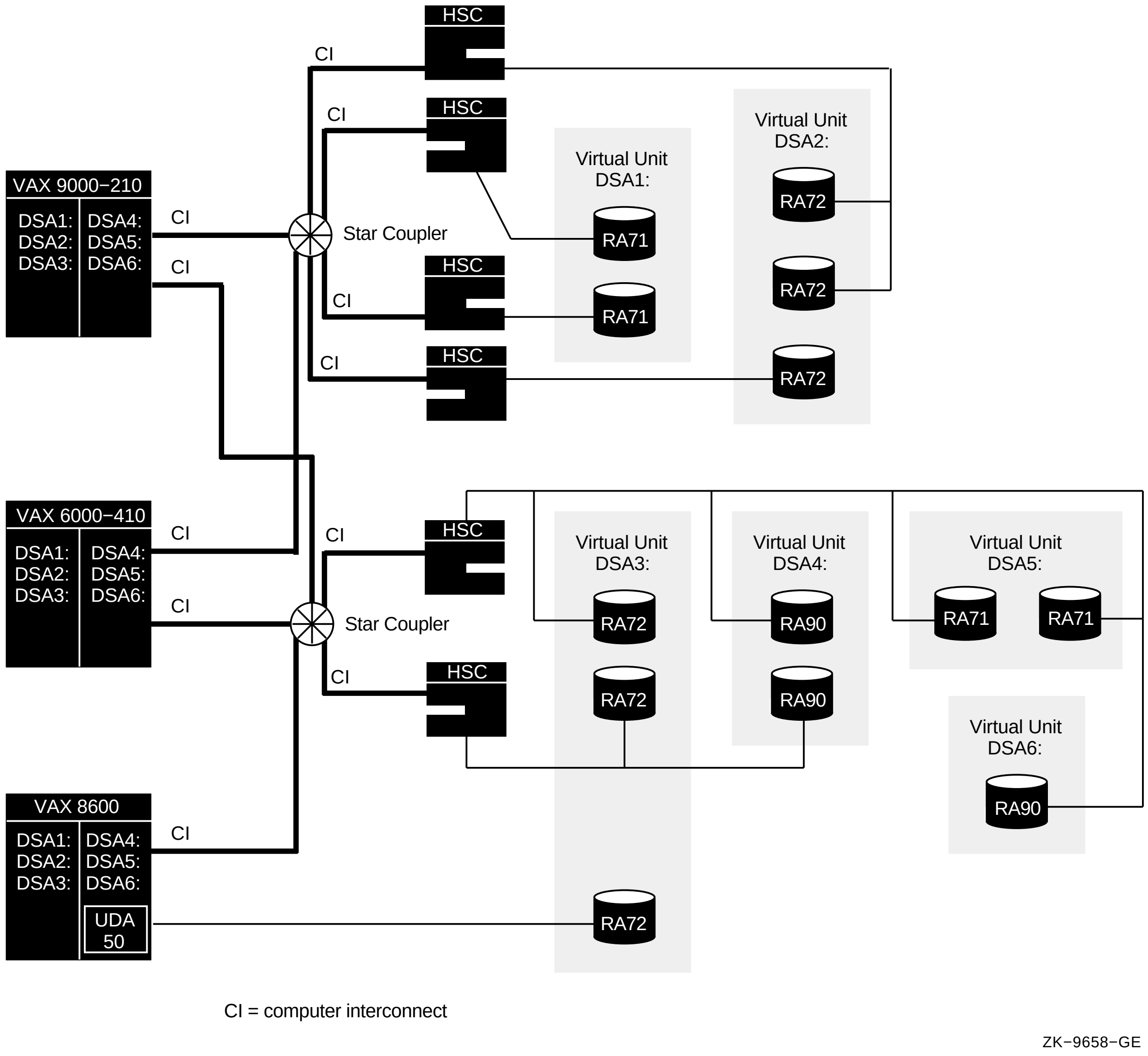


Figure 8–9 illustrates how shadowed disks can be located anywhere throughout a VAXcluster system. The figure presents a VAXcluster system with three VAX processors, multiple HSC controllers, and multiple shadow set members that are accessible by any processor. The shadow sets are accessible with three processors, with two processors, and, in some cases, with only one processor operating. The exception is if the VAX 9000 and the VAX 6000 processors fail, leaving only the VAX 8600 running. In this case, access to the secondary star coupler is lost, preventing access to the DSA1 and DSA2 shadow sets. Note that Figure 8–9 also configures shadow set members on different star couplers.

Figure 8–9 Configuration of a Shadow Set (With Multiple Star Couplers and Multiple HSC Controllers)



8.2.5 Disk Striping and Data Parity

OpenVMS supports disk striping and data parity on disks connected to AXP and VAX systems. Digital offers these performance and dependability features in a variety of hardware and software based products. Ask your Digital representative to supply you with the latest product descriptions and options. The most recent version of *The Digital Guide to RAID Storage Technology* is a useful document to read, and can be obtained from your Digital representative or by calling 1-800-344-4825 (1-800-DIGITAL). Also request the latest Software Product Description (SPD) document for StorageWorks RAID Software for OpenVMS.

Disk striping enhances performance by spreading data across multiple drives. User data is broken into segments called chunks. The relationship between chunk size and average request size determines whether striping maximizes performance for request rate or data rate. The system manager sets chunk size based on application requirements. If chunk size is set large relative to average request size, all of the drives may be able to execute different requests simultaneously. This load balancing is especially beneficial in applications in which there are large numbers of requests.

If chunk size is smaller than the average request size, multiple drives in a stripe set can participate in each request, thereby increasing data rate. This is valuable with large request sizes because data transfer time is a significant portion of total data access time.

Load balancing is an automatic outcome of striping. Without striping, frequently accessed data may be concentrated on a single drive, and that drive can become an I/O bottleneck. When striping spreads data among several drives, the I/O workload is balanced across several drives and total system performance benefits.

Disk striping alone does not provide data availability features and in fact makes data more vulnerable to outages. With disk striping alone, the loss of one disk drive in the stripe set can result in the unavailability of data across all the disks that comprise the stripe set. You can shadow striped disks to provide data availability, and you can stripe data in an existing shadow set.

A second way to provide data redundancy is with **data parity** features, which allow for the reconstruction of the original data if a drive fails. Stored in an array, this information makes it possible to lose a physical drive without losing access to the data stored on it.

Disk striping features can be combined with data parity features.

Compared to shadowing, parity reduces the cost of data redundancy at the price of reduced subssystem performance if a drive fails. Parity allows data reconstruction, but the speed of reconstruction depends on the system workload and the reconstruction algorithm used.

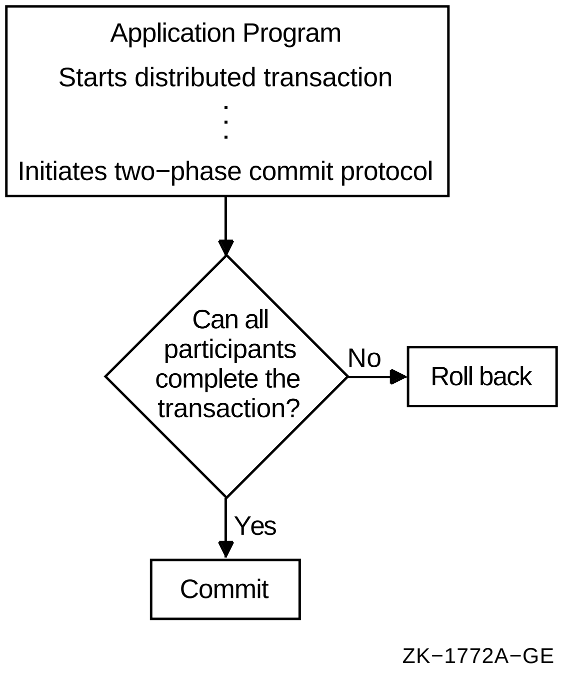
8.2.6 DECdtm Services and Two-Phase Commit Protocol

An efficient synchronization mechanism built into OpenVMS is the set of DECdtm services. By enabling your programmers to construct distributed applications that communicate via a two-phase commit protocol, the DECdtm services provide a high degree of robustness.

Managers of distributed applications need automated confirmations that a single transaction spanning multiple RMS files or databases in local or remote locations is guaranteed to complete as a single unit or not at all. A two-phase commit protocol is needed to ensure that all identified data resources affected by the work of the transaction are accessible and operational. Successful return calls must be received by the originator (program) from all the resources before the transaction commit sequence is considered valid; otherwise, the transaction is declared incomplete and the entire work of the transaction is rolled back.

DECdtm services, part of OpenVMS, provide system services that demarcate and coordinate distributed transactions. By using a two-phase commit protocol, these services ensure consistent execution of distributed transaction on the OpenVMS operating system. Figure 8–10 illustrates the two-phase commit protocol.

Figure 8–10 Two-Phase Commit Protocol for a Distributed Transaction


Application Program
Starts distributed transaction
Initiates two−phase commit protocol
Can all participants complete the transaction?
No
Roll back
Yes
Commit
ZK−1772A−GE

8.2.7 Disk Defragmentation Applications

OpenVMS provides a **movefile** subfunction that allows application developers to create their own disk defragmentation applications. Also, Digital provides a product called the DEC File Optimizer for OpenVMS, which uses the OpenVMS **movefile** subfunction to defragment files on a disk. Refer to Section 9.6.2 in this handbook for summary information about the **movefile** subfunction and the DEC File Optimizer for OpenVMS product. See the *DEC File Optimizer for OpenVMS Guide to Operations* for detailed information about this product.

8.2.8 RMS Journaling

RMS journaling is a data integrity tool that can help to protect your data in RMS files from being lost or becoming inconsistent. There are three types of RMS journaling that you can use. Each protects against a different type of potential problem:

- **After-image journaling** lets you redo the modifications that were applied to an RMS data file. After-image journaling protects against hardware failures such as a disk head crash. It also allows you to recover RMS files that are inadvertently lost or deleted.
- **Before-image journaling** lets you undo modifications that were applied to an RMS data file. Before-image journaling protects against bad or erroneous data (that might be caused by communications line noise or a data entry operator's error) by allowing you to return an RMS data file to a previous state, before the bad or erroneous data was introduced.
- **Recovery unit journaling** lets you define one or more sets of RMS operations within an application that must either be done in their entirety or not at all. Recovery unit journaling protects against data becoming inconsistent due to an interruption (for example, a system crash) during the execution of an application or due to an error detected by an application.

You can use one, two, or all three journaling types in your RMS application. RMS journaling can be used for any RMS file that is updated. Note that journaling is not applicable to non-RMS file operations, nor is it applicable to RMS files that are rewritten (with a new version number such as text files that are modified by a text editor) rather than updated in place. For example, RMS journaling would be appropriate for the data file SALES.DAT;1 which, after an update by an application program, continues to have the file specification SALES.DAT;1. The use of RMS journaling would not, however, be appropriate to the text file REPORT.TXT;1 which, after an update by a text editor, would have the file specification REPORT.TXT;2.

Journaling is applied on a file-by-file basis not on an application basis. For any RMS file, you can designate (or **mark**) the file for after-image, before-image, recovery unit journaling, or any combination of them. When a file is marked for journaling, journaling is performed every time the file is updated. Within any application, you can use any combination of files that are marked for different types of journaling along with files that have not been marked for journaling.

See the *RMS Journaling for OpenVMS Manual* for detailed information about using the RMS journaling feature to protect the integrity of your applications using RMS files.

8.2.9 OpenVMS Queue Manager Failover Capabilities

OpenVMS provides a single queue manager process that acts as a clusterwide server, accessing the queue database for all processes in a VMScluster environment. The centralized design enables the queue manager process to fail over to another node if the node on which it is running leaves the VMScluster. In terms of dependability, the most visible benefit of this design feature is that your users' batch and print jobs can survive node crashes when the users operate in a VMScluster environment.

See Section 9.7 for more information.

8.3 Writing Predictable, Dependable Code

Software problems differ from hardware problems in that the software does not consist of physical parts that degrade or deteriorate with time. However, hardware problems can cause corruptions to data that is under the control of the software and can cause changes to the environment in which the software is used.

You can judge the dependability of software by its ability to:

- Continue to produce correct end results while correcting for software errors
- Recover or restore application data and resume processing
- Allow fixes and upgrades for timely repair and maintainance of the software
- Minimize the impact of software faults on other applications or on the operating system
- Allow easy installation and, once installed, allow adjustments to the installation to make the software more suitable for its particular environment

Software maintenance accounts for a large percentage of the lifecycle cost and manpower effort. Repair and maintenance of software components can be complicated if there are many people involved such as your service vendor, the software supplier, your own staff, and perhaps contract programmers as well. Efficient maintenance relies on the documentation that you keep during the development and evolution of your software. Comprehensive records help facilitate problem analysis when software maintenance is required. The following sections discuss potential software failures and their impact on the dependability of your computing system.

8.3.1 Avoiding Errors During Software Specification

Errors made while specifying the software design are probably the most abstract reasons for software failures. During software specification, errors are often made due to erroneous assumptions, erroneous deductions, and communication errors. In some cases, software mismatches can result when the software design does not account for the fact that various departmental "cultures" within the same organization have different ways of doing business.

Because errors made during software specification affect the suitability of the software to your organization, the specification process is an important part of dependability. The system users may not only come to depend on the computing resources and expect that the resources are always available to them, but also require that the computing resources perform the job in a familiar manner.

The way to avoid errors during software specification is through comprehensive business analysis and requirements gathering (see Chapter 2), as well as techniques such as providing prototypes and application walkthroughs. These techniques enable your computer users to give early feedback on application function and user interface specification decisions.

In addition, there is no substitute for program testing by users who are representative of those who will use the software in a production environment. Whenever possible, have the test programs use live data in parallel with programs in production and compare the results between the two environments.

8.3.2 Avoiding Errors in Design and Implementation

Design and implementation errors result from typing mistakes or having incomplete (or incorrect) knowledge of the applicable software environment. Many times, poor software design results when the business problem's complexity exceeds the abilities or experience levels of the programmers. Most programmers consider that doing an excellent job in software design and implementation is all that is necessary for high quality software. However, for truly dependable software, you must also design the software for future growth, for robustness necessary to survive disasters, to survive frequent modification, and to work reliably with other software components in the system.

Your development process has the greatest impact on the ability to produce applications free of defects. A wide variety of computer industry literature is available on the topic of building a reliable software development process. See Appendix B for information about *The Digital Guide to Software Development*, *Managing the Software Process*, and other references.

8.3.3 Predicting Future Software Requirements

As the business requirements grow, the software must be able to expand in order to meet the increasing computing demands. How well the software is able to adapt to future business needs is known as its **scalability**. Although the software designer must do his or her best to foretell the computing needs of the future, even the most experienced programmers fail to predict the future precisely. A truism of computing is that system usage always expands. If the workload on a particular system kernel becomes too great, most sites either acquire a more powerful kernel or add more kernels. Either way, the scale of the business solution has grown and it can soon exceed even the designer's most radical predictions.

When the software fails to scale gracefully, the most-noticeable symptoms usually are in the area of performance problems. Examples of poor scalability include:

- Frequently accessed disk records become areas of contention
- Linear table searches become compute-bound
- An application control program becomes unwieldy when it must control many more objects than its user interface was designed to handle
- Insufficient capacity of communication paths

8.3.4 Surviving External Failures

Although you could say that the inability of software to survive external failures is both a specification error and a design error, in the context of this discussion, the lack of robust software results from inadequate goals.

There is a great deal of existing software that conforms very well to specifications. Such software is well-designed, does the right job, and executes its tasks efficiently. However, even well-designed software can be more vulnerable to external events than necessary. Software should be able to withstand adverse events whether or not the events are outside the control of the software.

For example, long-running programs can be made robust enough to survive system outages without losing their entire investment in processing. Programs can be coded to journal their internal processing state out to disk storage at regular intervals, and in the event of a failure to include a mechanism to restart processing from the last checkpoint. Electronic circuit (or weather) simulation programs are obvious candidates for this technique.

Batch jobs that have multiple steps (DCL commands) can be designed to automatically restart, at points other than the beginning of the job after a system failure. See Section 9.7 for details.

A wide variety of applications can gain higher levels of dependability by utilizing transaction processing monitors. You can protect applications from many types of system kernel, network, or data storage failures through the use of queuing and failover capabilities. You can build these features into your application code or have your programs execute within a Digital transaction processing (DECtp) environment. DECtp products, including VAX ACMS, protect their applications from a large number of external failures. See the DECtp documentation for more information on Digital transaction processing environments.

8.3.5 Providing for Software Evolution

One way to reduce the need for frequent modifications to your applications is to design them with flexibility. For example, if your staff is always changing and expanding the set of printers that are connected to your system, having to change source code to support the new printer setup would be a constant headache. If you design the software programs to provide access to any printer configured (with perhaps a user-customizable subset), then your personnel could have greater flexibility with lower risk.

8.3.6 Managing Systems Integration

Besides software concerns, systems integration combines hardware, personnel, cultural factors, and other factors. However, it is the conflict between mismatched pieces of software that very often prevents a computing resource from providing adequate service.

Unless your company has delegated the total responsibility for your computing system to a service vendor, you or someone on your staff must assume the role of systems integrator. This involves tracking the software configuration (product, version, date, and other factors) of your system and implementing sufficient testing (under load) to verify that your particular set of software components works together reliably. Your systems integration job is made more difficult if you must merge software that originated from your hardware vendor with software that your programming staff wrote, as well as other vendor's software.

Because of different interpretations of interfaces, specifications and protocols, and different assumptions about behavioral characteristics, even if all the components are themselves free of defects, there is still the potential for integration problems. Therefore, it is very important for you to acquire only very high quality software to make the validation process manageable.

Likewise, because of the multiple sources of code, good record keeping practices (on software configurations and on problem tracking) can be invaluable in connecting system perturbations with the onset of problems. The data might not provide the final diagnosis but it would provide a place for the analysis to start.

8.4 Prototyping Applications to Build in Dependability

You develop applications based on the requirements collected from the users of the application. Some users know what they want and can communicate their needs accurately to a designer or analyst. Other users are not sure what they want or are unable to communicate their needs. To check your application analysis, you might protoype some key applications.

Walk a user through the prototype version of the application to see if you have correctly understood the way the user carries on a business task. Based on the user's reactions, you might need to change the application design. It is much easier to correct an error based on misinformation at this early stage, however, than when the system is already in production.

Several good prototyping products are available:

- DEC Visual User Interface Tool (VUIT) software

  DEC VUIT™ is an interactive WYSIWYG-style (What-You-See-Is-What-You-Get) editor for building Motif® application interfaces. It is intended for developers experienced in constructing interfaces using Motif or similar windowing systems.

  DEC VUIT provides an environment that supports rapid development of graphic user interfaces that are compliant with OSF/Motif®. Prototype interfaces can be tested, quickly modified based on end-user input, and fed directly into the development stream. Because DEC VUIT generates industry-standard OSF/Motif UIL, not a proprietary presentation description language, the interface built using DEC VUIT is portable to any other platform that supports the OSF/Motif environment.

  DEC VUIT enables developers to build the interface visually instead of writing code. During editing, it uses inherent knowledge of widget properties and parent-child relationships to reduce run-time errors.

- DECforms software

  DECforms offers application developers a set of software development tools and a run-time environment for implementing powerful yet user-friendly human interfaces. DECforms supports the full range of VT100, VT200™, VT300™ , and VT420™ series terminals and compatible terminal emulators on PCs and workstations. DECforms integrates with VAX ACMS to provide forms processing capabilities in transaction processing environments.

- VAX DATATRIEVE™ software

  You can use VAX DATATRIEVE to define a record, a domain, Rdb databases, DBMS databases, and RMS files using ADT, a dialog-driven interface for data definitions. Then, develop interactive procedures using the VAX DATATRIEVE data manipulation facility. See the *DEC DATATRIEVE User's Guide* for more information.

- DEC RALLY software

  Use DEC RALLY to prototype applications that use DEC Rdb or RMS files and check the logical design of your database. DEC RALLY includes menu-driven tools that let you quickly develop an application prototype, including a form interface for data entry. An advantage of using DEC RALLY is that its prototype can be enhanced to become the final application, with the forms and data access combined into one. See the *Developing DEC RALLY Applications* for more information.

- DECquery™ software

  DECquery is a personal computer (PC) or Macintosh® based data retrieval system that accesses DEC Rdb databases.

- DECdecision software

DECdecision is an intergrated spreadsheet, database query and update interface, and graph creation package that is CDA™ compliant. DECdecision allows people who are not programmers to create database applications.

You can also prototype the logical design of your database using interactive utilities: SQL for DEC Rdb databases, and DBQ for DEC DBMS databases. Although these interactive utilities do not model data entry, they can present output data for user verification. If you implement your production database using DEC Rdb, the SQL statements in your prototype are very close to the statements programmers need to specify in application source code. Use DBQ to prototype transactions if you use DEC DBMS as your database manager.

8.5 Testing Applications to Verify Application Dependability

Each time you create or modify an application, it is possible that the application will run differently than expected. These unexpected changes are impossible to predict and often difficult to uncover.

The best method for detecting such changes is to catch them as soon as they occur so that you can identify which modification caused the change. Testing should also adequately check the application's dependability when it is stressed by an unusually heavy transaction load. The following list provides guidelines for testing applications for dependability:

- Plan to test applications as part of the development process.

  To detect defects efficiently, you need to continuously test the application and to compare its actions against expected or past results. Only by carefully planning and testing the system modifications can you ensure that your application modules are stable and dependable. Be prepared to allocate test system design and implementation time, as well as system resources, in order to conduct adequate tests for dependability and performance.

- Keep the test environment separate from the production environment.

  For accurate test results, test the application software in the same or a similar environment as the one where you intend to run the application. For example, you might use two identical system roots and system logicals on both systems and run actual production I/O data on both the test and production systems.

  Note that the test system should be a strictly remote test system so that any failures that might occur do not affect the production system. If it is not possible to use actual production data to test the application, use the DEC Test Manager to simulate users entering data into applications and printers to simulate the recipients of outgoing I/O. Through these types of elaborate test environments, test results can be compared with actual run-time results to validate the dependability of the new or changed software.

- Use both reasonable and unreasonable test data.

  Using actual production data is, obviously, critical as you verify application logic and system reliability. You should also throw in data that is unacceptable to the application so that you can catch software errors in the application that would show up only when users or other system components behave in unpredictable ways. That is, you should include thorough testing of boundary conditions. Problems that are exhibited at the limits of the data set size or input values are common.

The goal is to discover if the application will respond in an acceptable manner to all possible input, however unlikely. Usually, extra effort spent trying to break the system during testing is paid back many times over from the absence of problems during real production.

- Manage your configurations carefully during the testing phase.

  Practice good experimental design practices, such as controlling as many variables as possible and changing them only after much deliberation for each test. The hardware and software configuration of your system represents a very large collection of variables that could change. If you also consider network topology and status, you further add to the number of outside influences that could affect test results.

  If you strictly control and record all the details of your system's complete configuration during testing, it will be easier to diagnose problems in the code.

- Document the test results completely.

  Efficient testing relies on the documentation that you keep during the development and evolution of your software. Adding comprehensive testing records to that documentation helps facilitate problem analysis when software maintenance is required. The benefit of maintaining detailed documentation on an ongoing basis is that much of the system management wisdom acquired in the testing and maintenance process is naturally integrated into the computing system on an ongoing basis.

- Provide the ability to quickly restore the production system to its preupgrade status.

  If possible, avoid upgrading any software modules without having a procedure in place to immediately restore the system to its prior, stable operating state. Should a problem arise with the newly installed application software, you can immediately restore the computing system to its previous dependable state.

  When it is not feasible to restore the production environment to its original stable condition, the importance of exhaustively testing the new software on production-like testbeds becomes even more critical.

Finally, there are many useful products for testing your application software. Although the following list is not all inclusive, it suggests several products useful for application testing:

- The OpenVMS debugger
- DEC Test Manager
- DEC Performance and Coverage Analyzer (DEC PCA)
- VAX Performance Advisor™
- POLYCENTER Performance Solution
- DECtrace™
- RdbExpert

8.6 Dependability Features of Application Software

This section describes the dependability features of:

- Database applications; specifically, DEC Rdb. Also, related products like RdbExpert and DECtrace. See Section 8.6.1.
- Products that define the application form; specifically, DECforms, a Digital implementation of a proposed ANSI and ISO standard for a Form Interface Management System (FIMS). See Section 8.6.2.
- Transaction processing (TP) monitors; specifically, VAX ACMS. See Section 8.6.3.

8.6.1 Building Dependable Database Applications

In the OpenVMS based environment, DEC Rdb provides features that contribute to the continuous processing of business functions. These features include:

- Failover in VMScluster environments. See Section 8.6.1.1.
- Full and incremental backups of selected portions of a database or of an entire database, while there are active users running update transactions and modifying data. See Section 8.6.1.2.
- Data integrity and data consistency when a single transaction works with multiple Rdb databases, using a two-phase commit protocol. See Section 8.6.1.3.
- Automatic cleanup of Rdb databases. See Section 8.6.1.4.
- Modification of certain physical database components while there are active users working with the database. See Section 8.6.1.5.
- Security management commands. See Section 8.6.1.6.
- Support for using DECtrace and RdbExpert with Rdb applications. See Section 8.6.1.7.

For details about the DEC Rdb dependability features described in the following sections, refer to the DEC Rdb documentation set. In particular, see the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning*.

8.6.1.1 Database Failover in VMSclusters

DEC Rdb in a VMScluster environment allows access to a database from many concurrent users on multiple nodes. DEC Rdb automatically recovers your database if an AXP node or VAX node in your VMScluster system fails. It also provides optional after-image journaling to further protect the integrity of your database.

In a properly configured VMScluster environment, DEC Rdb can give you almost constant availability to your database.

How DEC Rdb Databases Work in VMSclusters

The OpenVMS distributed lock manager provides synchronization of resources across the VMScluster. DEC Rdb uses the OpenVMS distributed lock manager to synchronize clusterwide updates to the database files, to initiate the automatic recovery process when an AXP or VAX node fails, and to coordinate concurrent updates to the same database from processes running on different AXP or VAX nodes in the VMScluster.

Along with the increased availability of data residing in DEC Rdb databases in VMScluster environments comes increased database administrator (DBA) planning tasks and responsibilities. In a VMScluster environment, the versions of DEC Rdb on all AXP nodes or VAX nodes in the cluster must be the same and all database files must be on devices accessible to all AXP nodes or VAX nodes that share the database.

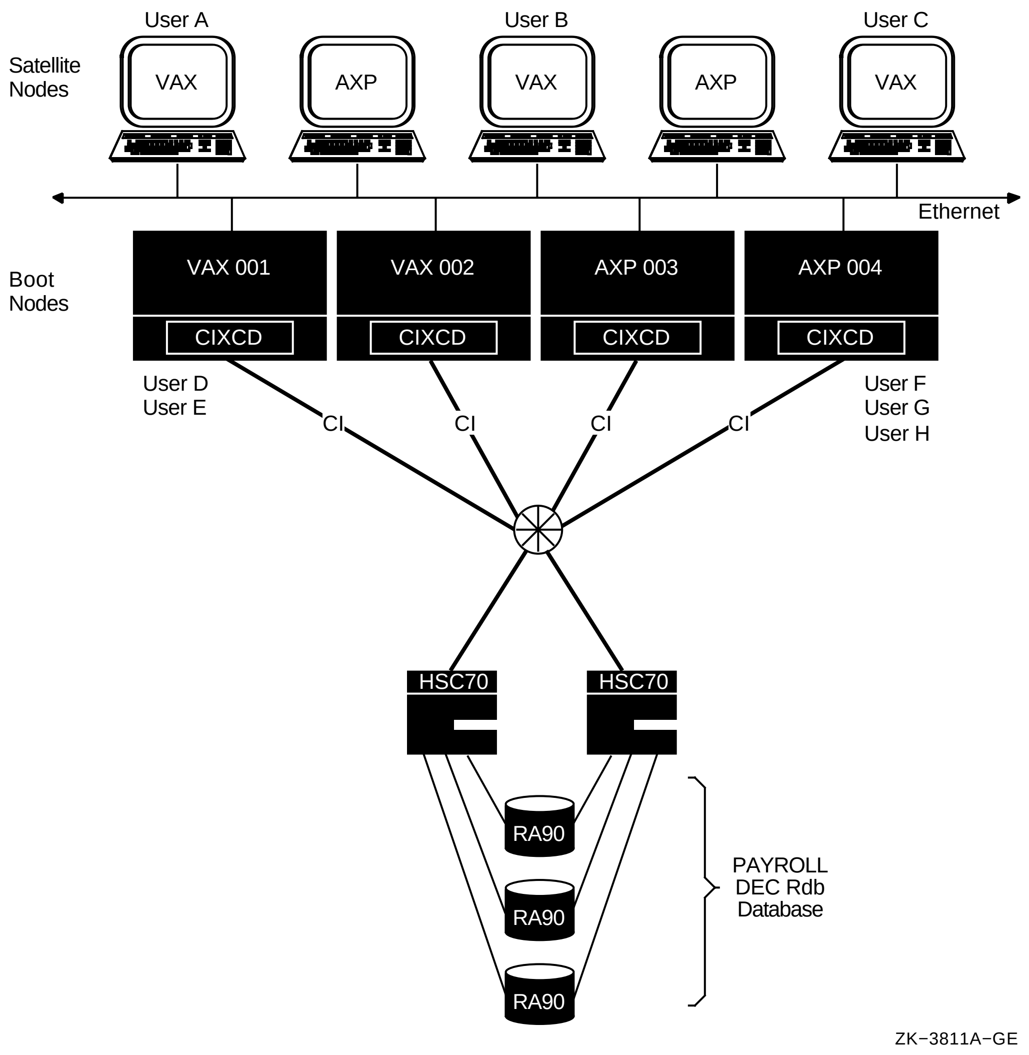
In particular, the recovery-unit journal (RUJ) files must be accessible for automatic recovery to take place if one of the nodes fails. Each process accessing a DEC Rdb database in READ_WRITE transaction mode has an RUJ file, which contains records from the database as they appeared before update operations started by the current transaction. Automatic database recovery operations return the preupdate records from each RUJ file of processes that were interrupted by a resource failure (before the processes with active transactions had a chance to commit). This ensures that incomplete transactions do not remain in the database.

In a VMScluster environment, an RDMS_MONITOR monitor process must be running on each node from which users will access any DEC Rdb database. Communication between the DEC Rdb monitor process on each node is handled through the OpenVMS distributed lock manager. The monitors on each AXP or VAX node use locks synchronized by the OpenVMS distributed lock manager to signal between processes that a significant event has occurred. A **significant event** in a VMScluster is when a process has terminated abnormally or a node in the VMScluster has failed.

When an AXP or VAX node fails in a VMScluster environment, the OpenVMS distributed lock manager alerts DEC Rdb monitor processes on other nodes on which there are users of the same database. One of these monitor processes then initiates the DEC Rdb recovery procedure to roll back the transactions that were in progress for users on the failed node.

Figure 8–11 illustrates a VMScluster configuration in which there are eight active DEC Rdb users of the PAYROLL database when VAX001 fails.

Figure 8–11 Failover and Recovery Process for DEC Rdb Users in VMScluster



In Figure 8–11, when the VAX001 node fails:

1. The active database transactions of User D and User E are interrupted and a software bit is set in the database that the work performed by these two users needs to be rolled back.
2. A DEC Rdb monitoring process on one of the surviving nodes will trigger the recovery operations in recovery processes on behalf of User D and User E.
3. In the recovery operation separate DEC Rdb recovery processes read from the separate, process-specific RUJ files of User D and User E and use this data to reconstruct the preupdate versions of records affected by User D's and User E's transactions.

   During this recovery operation, Users A, B, C, F, G, and H on the other VMScluster nodes may perceive a pause in their database activity. However, their transactions to the same PAYROLL database remain active and viable. When the recovery operation completes, the users resume their transactions without any ill effects on the data integrity or continuity.

4. After DEC Rdb recovers the PAYROLL database, users who were on a failed VAX node can log in to another AXP or VAX node in the VMScluster and continue using the database. Therefore, you should ensure that users have accounts on surviving nodes or that your VMScluster configuration uses a common SYSUAF.DAT file. All VMScluster nodes that access the same DEC Rdb database should use the same, or coordinated, SYSUAF and Rights database. The DEC Rdb protection scheme is tied to the AXP or VAX system on which the database was created.

Where to Place DEC Rdb Files in VMSclusters

The following guidelines can help you decide where to place the database files:

- Database root (RDB), storage area (RDA), and snapshot (SNP) files

  The database root, storage area, and snapshot files must be accessible from every node that intends to access the database. DEC Rdb must be able to create and to maintain VMScluster distributed root file access.

- Recovery-unit journal (RUJ) files

  DEC Rdb must be able to complete its automatic recovery procedure should a node fail. Recovery can complete only if all RUJ files are accessible from every node that accesses the database. To ensure that all RUJ files are accessible, follow these rules:

  – Define the logical name RDMS$RUJ to point to a directory on a device that has clusterwide accessibility and has more than one access path, if possible. The logical may be defined in the system table of each process.

    Make sure that there is no version limit on the directory. Also, set and test the protection on the directory so that everyone who needs to access the directory can indeed access it.

  – Make certain that if you permit RUJ files to reside in the default directory named [RDM$RUJ] (on the device where the user's SYS$LOGIN is defined), the [RDM$RUJ] directory's device is on a cluster-accessible disk.

  – Do not place a user's RUJ file on a disk on a remote node of a local area VMScluster.

  – To ensure good performance, place journal files on a device other than the ones on which the root file (.RDB) and storage area files (.RDA) are located.

- After-image journal (AIJ) file

  The after-image journal (AIJ) file contains all updated records by all users of the database. The beginning and ending time period of each AIJ file is defined by the DBA. The AIJ file must be accessible from every node that accesses the database. This ensures that all processes that access the database will be able to write to the AIJ file. Never place the AIJ file on the same disk as the one(s) holding the root file (.RDB) or storage area files (.RDA). In the event of a disk failure affecting the .RDB or .RDA files, you will want to apply the surviving .AIJ file to a backup copy of the database.

  Also note that while having an AIJ for a database is optional, it is highly recommended for critical applications. The decision to use or not use AIJ with a particular database depends on the value placed on the data and the ability to recover lost work since the last backup.

With DEC Rdb multifile databases, use the RMU/SHOW STATISTICS command and the MONITOR DISK command during normal and peak database activity periods to evaluate the I/O rates on each disk. Use the RMU/MOVE_AREA command to relocate any storage area files that currently reside on disks with excessive I/O rates.

Note

Never place journal files on the same disk as the database or RMS file they are protecting.

Also note that if your organization uses the CDD/Repository software, the data repository files should be placed on a shareable cluster disk that is always accessible to users from any node in the VMScluster configuration.

Minimizing Impact of Component Failure on Database Access

To ensure continuous database access by surviving nodes when a node in the VMScluster configuration fails, do not place any DEC Rdb files on disk devices that are cluster-mounted on single-ported, MSCP-served disks. If you place any DEC Rdb files on single-ported, MSCP-served disks, and the AXP or VAX processor to which the disks are connected fails, the DEC Rdb recovery procedure cannot access those files. Activity on the database ceases until the failed AXP or VAX processor can be restarted.

To reduce the chances of losing access to your database due to a system component failure, consider the following facts:

- If you place your files on disks connected to a pair of HSC or HSJ devices, you lose access to your database files only when both HSCs or HSJs fail at the same time. When one HSC or HSJ fails, access to the disks continues through the alternate path. (Remember to use the common allocation class name when you define your database files, as explained in the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning*.
- If you place your files on dual-pathed MSCP-served disks, you lose access to your database only when both nodes fail at the same time.
- If you place your files on a single-pathed HSC or HSJ disk, you lose access to your database when that HSC or HSJ fails.
- If you place your files on a single-pathed disk that is not an HSC or HSJ, you lose access to your database when that node fails.
- In a local area VMScluster configuration, if one of the boot nodes fails, cluster operations will be suspended until the failed node rejoins the VMScluster. This condition is normal and ensures the integrity of shared VMScluster resources.

Refer to the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning* for details about the operation of DEC Rdb databases in VMScluster environments. See *VMScluster Systems for OpenVMS* for information about the different types of VMScluster configurations. Section 8.2.2 summarizes the VMScluster and application environment topologies as they relate to system dependability.

8.6.1.2 Backing Up Active DEC Rdb Databases

One reason for using a database management system like DEC Rdb is to protect data from hardware failures, software failures, and human errors. Two types of operations are necessary to protect the data in the database:

- Saving copies of the database at regular intervals
- Making copies of completed update transactions in an AIJ file

When you use both methods of protection, you can rebuild lost or corrupted databases. If anything happens to your database and you have maintained a regular schedule of backups, you can restore it to the state it was in when you last backed it up. If you keep an AIJ file for the database, you can apply AIJ entries to the restored database. The AIJ process reapplies database updates that occurred since the last backup operation and before the database corruption or loss.

DEC Rdb provides its own specialized backup tool that is based on the structure of a database (use of snapshot files) to achieve a high degree of availability of the database application during this type of maintenance operation.

Note

Do not use OpenVMS Backup utility with DEC Rdb files. Relying on OpenVMS Backup may compromise the reliability and availability of the database. Digital recommends using the RMU/BACKUP and RMU /RESTORE commands with DEC Rdb databases. See the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning* for details.

The DEC Rdb backup mechanism provides a number of features for DBAs and system managers who are supporting applications that require concurrent access to shared, critical data in continuous processing environments:

- Backup operations can occur while the database is open to active users (transactions). Special synchronization occurs with active update transactions. This feature is called **online backup**.
- You can perform incremental or full backups. Incremental backups reduce backup time and ease maintenance by reducing the amount of data that needs to be backed up. A full RMU/BACKUP command backs up the RDB file and all RDA files in all the storage areas into a single database backup (RBF) file. An incremental backup backs up the RDB file and all database pages that have been changed since the most recent full backup into a single RBF file. Full backups and incremental backups do not copy empty database pages or SPAM pages, thereby saving space in the backup file and minimizing the elapsed time of the RMU backup and restore operations.
- A DEC Rdb database can be divided into physical data portions, defined by the DBA, called **storage area files**. DEC Rdb lets you specify which storage area files you want to back up.
- You can run multiple magnetic tape drives concurrently during the backup operation in a feature called multithreaded backup. This feature can significantly decrease the amount of time needed to back up large databases. When the sum of the peak data rates of the tape drives exceeds the computer interconnect (CI) port bandwidth, drive performance drops significantly. Digital recommends that the DBA experiment with the number of concurrent

tape drives that can be used during a multithreaded backup operation and that can result in a faster backup elapsed time (before overwhelming the CI port bandwidth). Note that the CI port bandwidth is improved with newer or additional CI ports.

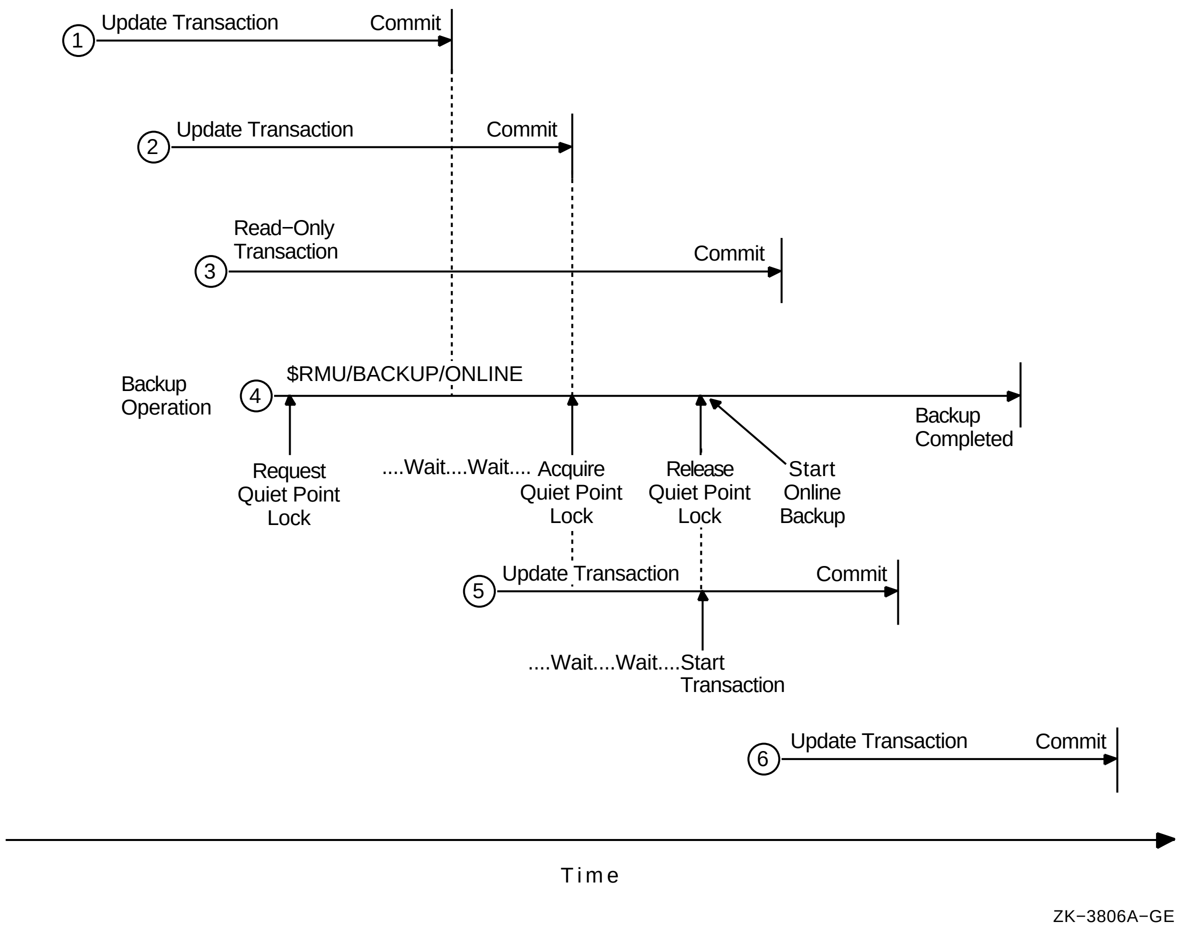
Online Database Backup Operation

You can perform a full or incremental backup of your database on line, without closing the database or denying access to users. When the online backup command is entered by the DBA, DEC Rdb requests a “quiet point” lock and waits for all active READ_WRITE transactions to complete. The online RMU backup will proceed when the database reaches a quiet point; that is, a moment when there are no active READ_WRITE transactions.

Once the online DEC Rdb operation has started, subsequent READ_WRITE users can invoke (attach to) the database. DEC Rdb will wait until the online backup operation releases its quiet point lock before actually allowing these subsequent READ_WRITE transactions to start their updating activity. This technique ensures that the data backed up represents a consistent view of the database at the moment of the online backup operation. Through the use of internal transaction sequence numbers (TSNs), transactions that performed work after the online backup requested its quiet point lock will be captured in the next backup operation.

The time line in Figure 8–12 shows how DEC Rdb performs the online backup operation when there are active users.

Figure 8–12 Coordination of Online Database Backups



The numbers in Figure 8–12 correspond to the numbered explanations in the following list. The following transactions occur for the same DEC Rdb database that is being accessed by five users and, at one point, a concurrent DEC Rdb online backup operation:

1 An update transaction starts. For example, this could be a READ_WRITE ... FOR PROTECTED READ transaction.

2 An update transaction starts. For example, this could be a READ_WRITE ... FOR EXCLUSIVE WRITE transaction.

3 A READ_ONLY transaction starts. The quiet point has no effect on READ_ONLY transactions, which have no effect on the online backup operation.

4 The DBA enters an online backup command, $ RMU/BACKUP/ONLINE [parameters]. RMU immediately requests a quiet point lock in EXCLUSIVE mode and then waits until all currently executing READ_WRITE transactions (numbers 1 and 2) complete (COMMIT or ROLLBACK). When transaction 1 completes, the online backup operation still must wait because transaction 2 is active. When transaction 2 completes, the backup operation acquires its quiet point lock.

Note

You can specify a LOCK_TIMEOUT option for the RMU online backup so that RMU will "give up" if it has not acquired the quiet point lock in a reasonable time (specified by the DBA). The LOCK_TIMEOUT option ensures that the culprit READ_WRITE transaction that is not completing within the defined reasonable time can be identified and dealt with by the DBA. Once the culprit READ_WRITE transaction is handled, the DBA can restart the online backup operation with hopefully less impact on cooperative READ_WRITE transactions.

---

5 At this point during the online backup operation's waiting period, a new READ_WRITE transaction "begins." For example, this could be a READ_WRITE...FOR SHARED WRITE transaction. However, because the internal transaction sequence number (TSN) of this new transaction will be greater than the TSN of the (still waiting) RMU/BACKUP/ONLINE transaction, transaction 5 will wait until the online backup operation releases its quiet point lock.

Any updates made by transaction 5 will not be reflected in this online backup operation. This behavior ensures the integrity of the backed up data; the next incremental or full backup operation will include any updates made by transaction 5.

Note that shortly after transaction 5 started, transaction 2 completed. When the final READ_WRITE transaction completed, the online backup operation acquires its requested quiet point lock.

Note

When the RMU online backup operation acquires the quiet point lock, RMU holds the lock for a very short time (typically, just seconds) while it acquires the database state and other database locks. None of these operations have longer than typical I/O delays associated with them.

When RMU releases the quiet point lock, the online backup operation actually begins.

Once the online backup operation releases its quiet point lock, transaction 5 no longer waits; transaction 5 proceeds as long as it does not take out any exclusive locks on data currently being backed up.

6 A READ_WRITE transaction starts after the online backup operation. For example, this could be a READ_WRITE...FOR PROTECTED WRITE transaction. This transaction 6 can proceed during the online backup operation, which has released its quiet point lock, as long as transaction 6 does not require any exclusive locks on data currently being backup up.

Again, any transactions that started after the quiet point lock was established by DEC Rdb will be captured (by means of an internally recorded TSN) in the next incremental or full backup operation, in the optional AIJ file, or both.

Large Database Backup Strategies

For large databases that require no interruptions or minimal interruptions, carefully plan your backup strategies and be conservative. With the DEC Rdb features, considerable time and resource savings are possible by carefully modifying your backup strategy to take advantage of the /INCLUDE and /EXCLUDE qualifiers in the RMU/BACKUP command.

This observation is especially true for large to very large databases and for databases that are mostly read-only. Explicit use of the /INCLUDE and /EXCLUDE qualifier results in a partial backup. This backup may be either full or incremental in the sense that either all the storage area pages or only those pages modified since the last complete full backup may be incorporated into the backup file. Because a backup file may be a partial backup, it is necessary to ensure that all storage areas of a database are restored and recovered when using the RMU/BACKUP and RMU/RESTORE commands to duplicate a database. That is, the synchronization of backups is essential when backup by storage area is performed.

It is absolutely critical that you develop a regular, frequent database backup schedule that meets the failure recovery requirements of your business. It is also critical that you develop, in advance of a data disaster, comprehensive restore strategies. To appreciate the impact on your business of lost data, try to estimate the cost on a time basis. If a disk failed today, what is the cost of losing one minute, one hour, one day, one week (and so on) of database transactions? Defining your requirements today may indeed save you from painful experiences in the future.

Refer to the RMU backup chapters in the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning* for detailed information on your backup and restore strategy options.

8.6.1.3 DEC Rdb Use of Two-Phase Commit Protocol

Some applications need to exist in a distributed environment to match business requirements. A distributed environment typically consists of many nodes that are geographically distant and connected by networking components. When an application program runs a single transaction that accesses multiple resources such as multiple DEC Rdb databases, RMS files, or both, the program needs assurances and confirmation that all updates can be made successfully to all the resources. If for any reason the updates cannot be made to all the resources, then assurances are necessary that no updates will be made to any of the resources.

To solve this technical problem, a two-phase commit protocol was defined. The two-phase commit protocol is a procedure where:

1. All the participants in a distributed transaction first agree to commit
2. On a signal from a coordinator, all the participants then commit or none of the participants commit

Figure 8–10 illustrates the two-phase commit protocol. The important point is that all of the participants will do the same thing. Either they all will commit or they all will abort (roll back).

DEC Rdb supports the two-phase commit protocol that is provided and coordinated by DECdtm services, which is part of OpenVMS.

Note

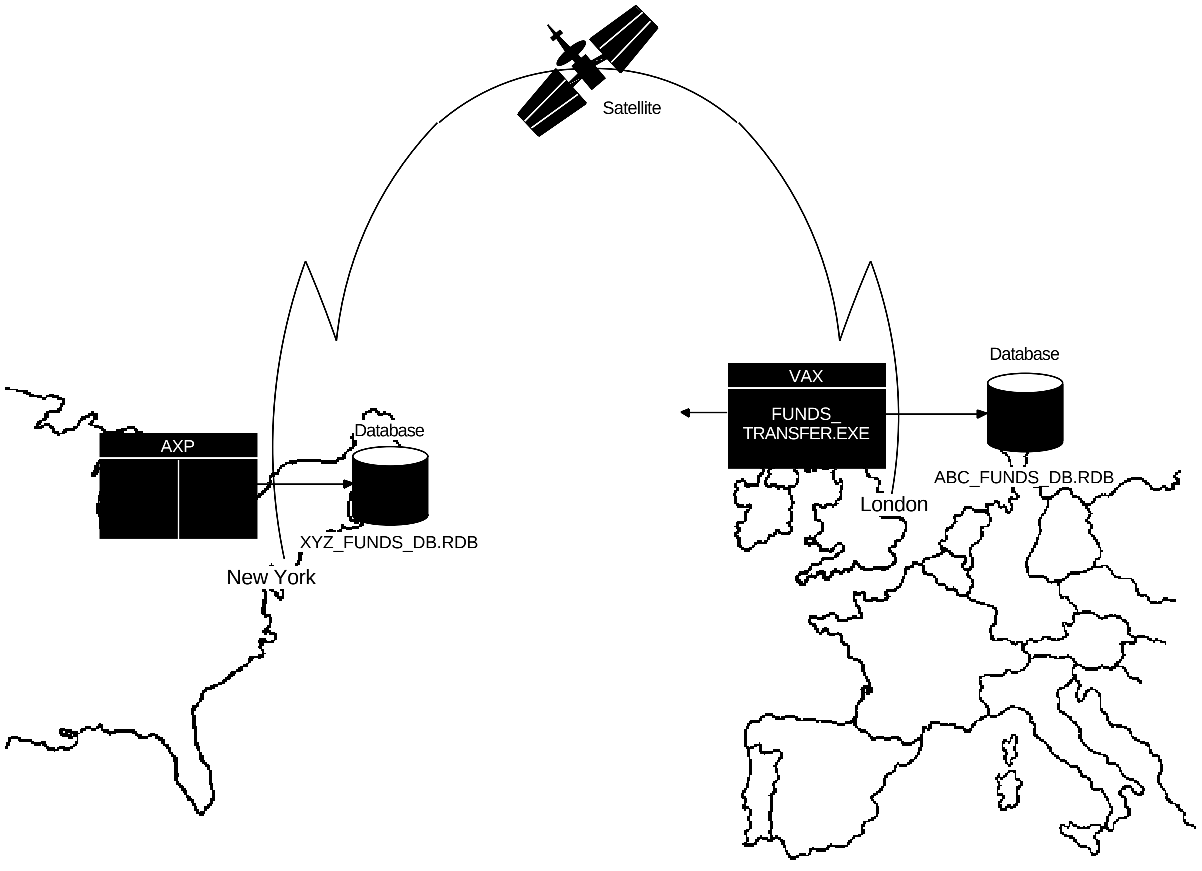
Refer to the *VAX Rdb/VMS Guide to Distributed Transactions* for information about using distributed transactions with DEC Rdb databases. See the DECdtm chapter in the *OpenVMS Programming Concepts Manual* for DECdtm programming information.

Transactions can span multiple DEC Rdb databases, DEC DBMS databases, or RMS files (not just DEC Rdb databases).

Using the two-phase commit protocol, you can divide a large DEC Rdb database into several smaller databases. Or you can create applications that access several different databases without compromising the integrity or consistency of your data.

For example, assume that the FUNDS_TRANSFER application shown in Figure 8–13 needs to withdraw $1,000.00 from the ABC Bank's DEC Rdb database in London and deposit the money in the XYZ Bank's DEC Rdb database in New York.

Figure 8–13 Two-Phase Commit for Funds Transfer Example



ZK−3807A−GE

In Figure 8–13, the ABC Bank customer's beginning account balance is $4500.00. From London, the FUNDS_TRANSFER application:

1. Attaches to the ABC_FUNDS_DB and the XZY_FUNDS_DB databases.

2. Starts a single update transaction that is declared to work with both databases.
3. Withdraws $1000.00 from the ABC customer’s account. The account balance is changed to $3500.00. The single update transaction is still active.
4. Starts the update to deposit $1000.00 into the specified XYZ account.

   Assume that at this point, the network connection from London to New York fails. The $3500.00 balance in the ABC Bank customer’s account is invalid because it cannot be immediately determined whether the electronic transfer completed to the XYZ Bank customer’s account.
5. Detects the error in attempting to reach the New York XYZ Bank. Even though the update (withdrawal) to the ABC Bank was successful, the entire work of the transaction was not successful. A directive to roll back all changes is made, restoring the ABC Bank customer’s account balance to $4500.00. The roll back ends the transaction. The application could continue attempts to perform the funds transfer and will commit safely only when it can be determined that the work with all resources was successful.

Note

When multiple resources are involved, the application must include code to respond reasonably to complex failure scenarios. The two-phase commit protocol will ensure the integrity of the data, but the application must ensure that the user’s task is eventually completed.

8.6.1.4 Automatic Cleanup of Databases

DEC Rdb software can perform the following automatic cleanup tasks while users are attached to a database:

- Creation, extension, or deletion of a user’s recovery-unit journal (RUJ) file as required
- Recovery of any aborted transactions
- Use of freed space in database storage area files and snapshot files
- Extension of any storage area files, as required
- Extension of after-image journal (AIJ) file, as required
- Change in lock granularity of a user’s transaction, as required
- Detection of deadlock situations between contending users for common resources
- Updates to approximate cardinality of tables and indexes, as appropriate

8.6.1.5 Online Restructuring of Database Characteristics and Definitions

Database administrators (DBAs) typically must manually adjust a database’s physical and logical definitions as demands on the database change over its lifecycle. In addition, a number of database administration tasks must be performed frequently to maintain the health of the database and the applications that depend on the database; for instance, performing incremental or full backup operations. In many database products, changing database characteristics and definitions and performing DBA maintenance tasks requires the DBA to shut down the database to its users while the offline database is serviced. These tasks disrupt business functions and often must be scheduled well in advance.

DEC Rdb permits DBAs to alter many database characteristics and definitions while there are active users bound to the database. The following list summarizes the online changes that are permitted with DEC Rdb databases:

- Performing online backup of the database, either full backup or incremental
- Performing online spooling of the AIJ file (that is, truncating and backing up contents of the AIJ file)
- Creating an online copy of a database
- Monitoring and recording performance statistics
- Displaying the contents of a database storage area (header, indexes, data areas)
- Changing the OPEN MODE of database
- Starting an RMU/VERIFY check of database integrity (this operation must wait for exclusive and batch-update transactions to complete)
- Updating metadata (definitions of data), including:
  - Creating, altering, or dropping collating sequences
  - Creating or dropping constraints
  - Creating, altering, or dropping domains, indexes, and tables
  - Granting or revoking protection
  - Creating or dropping trigger definitions
  - Creating, altering, or dropping storage maps (pointers to the file(s) in which particular data will be stored and the format the data will be stored)
  - Creating or dropping views (ad hoc logical representations of database tables)

The availability features in the previous list help the DBA maintain the database and tune the database's characteristics to improve performance. This degree of flexibility makes it easier for the DBA to change the database to match changing business needs. For the many functions listed in the previous list, it saves the DBA the trouble of having to:

- Schedule time for the database shutdown
- Provide adequate advance notice for all users (or software applications) of the database(s)
- Back up each database just prior to the change
- Close the database with an RMU command
- Perform the changes
- Verify and test the changes
- Open the database and announce its accessibility

At the same time, it is important for the DBA to be aware of the database administration functions that do require closing a DEC Rdb database. DEC Rdb database functions that must be performed **off line** (when there are no users attached to the database and the database is closed) include:

- Creating, altering, or dropping a database or a storage area

- Changing the following database characteristics:
  - The maximum number of database users
  - The maximum number of VMScluster nodes that can access the database
  - Space allocation characteristics such as extension values of the database

Note

The database can dynamically extend as required, but the default value of the size to extend can be altered by the DBA.

  - Database locking characteristics (whether automatic locking granularity is enabled or disabled)
  - The default number of database buffers per user
  - The default number of database recovery buffers
  - After-image journaling (change the file name used or the enable/disable option)
  - Usage of snapshot files (enabled or disabled; if enabled, whether immediate or deferred)
  - Allocation characteristics for the AIJ file
  - Allocation characteristics for a snapshot file
  - The requirement for a repository (CDD/Repository) to be used when metadata is changed
- Moving a storage area

Refer to the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning* for details about the online and offline DBA functions performed with DEC Rdb databases.

8.6.1.6 Database Security Impact on Application Dependability

Just as with an AXP node or a VAX node's physical and electronic security, a database is dependable only to the extent that its security requirements can be met. An application that can never fail can be protected by such measures as shadowed disks, journaling, online backups, and node failovers in a VMScluster environment. But if those steps are taken and yet the database is left unprotected against accidental and malicious attacks, then the application may be vulnerable and, consequently, may not be dependable enough.

DEC Rdb includes security features that comply with the U.S. Department of Defense Class C2 Security. C2 Security, as defined in the Department of Defense "Orange Book," provides controlled access protection. The security features in DEC Rdb are auditing capabilities and parameters that control access to database components.

DEC Rdb Access Control for C2 Security

Tables created with DEC Rdb have a default PUBLIC access of NONE. Default PUBLIC access to the database itself will be NONE also. You can modify this default by providing a DATABASE identifier of DEFAULT. The access given to this identifier will then be assigned to any newly created tables. You cannot attach to the database unless you have either the SELECT privilege (SQL) or READ privilege (RDO). You can assign the privilege using the GRANT statement in either SQL or RDO.

Note

Any user with BYPASS or SYSPRV privilege can circumvent database protection parameters defined by DEC Rdb users. Therefore, it is extremely important that your organization heed the documented warnings (including those in Chapter 9) and establish strict controls around the use of OpenVMS privileges.

Also, management of the database may require SYSPRV for the DBA. The database protections are stored within the database. If the database is inaccessible, the protections cannot be checked. An example of this is database AIJ recovery. If the database needs recovery, the database privileges cannot be assumed to be valid and thus SYSPRV is required of the DBA who will start the AIJ recovery operation.

DEC Rdb also provides role-oriented privileges, DBADM, OPERATOR, and SECURITY. Users with these privileges can override ACLs for some objects to perform certain system-level operations. DEC Rdb role-oriented privileges are limited to the database definition in which they are granted, whereas OpenVMS role-oriented privileges span all definitions on the system. These three privileges cannot override each other. For example, the DBADM privilege does not override the SECURITY privilege. For details on these privileges, see the *DEC Rdb SQL Reference Manual*.

DEC Rdb Auditing Capabilities

Auditing a database is done to identify attempts to compromise security. When you invoke the DEC Rdb security auditing feature, you can monitor and collect security audit records on many specific operations performed on a database. You can also send alarms to specified operator's terminals. This information can then be used to decide if and what further steps are necessary to make the database more secure.

DEC Rdb security auditing is modeled after the OpenVMS auditing model. Auditing can be done on a per database basis. Following are some examples of the commands that make up the DEC Rdb security auditing model:

- The RMU/SET AUDIT command enables DEC Rdb security auditing. When security auditing is enabled, DEC Rdb sends security alarm messages to terminals that have been enabled as security operators and makes entries in the database security audit journal whenever specified audit events are detected.
- The RMU/SHOW AUDIT command displays the set of security auditing characteristics that have been established with the RMU/SET AUDIT command.

- The RMU/LOAD AUDIT command allows you to load records from a DEC Rdb audit journal into a DEC Rdb table.

You can obtain information about the audit characteristics of a particular database by displaying the header of the database root file (.RDB).

8.6.1.7 Using DECtrace and RdbExpert with Database Applications

DECtrace collects and reports on event-based data gathered from DEC Rdb database applications. You can use DECtrace to collect detailed information about applications that use DEC Rdb. When you select a workload option in DECtrace, you can gather workload data from DEC Rdb database applications. This data can be imported into RdbExpert for use in generating an optimized physical design for your database. (DECtrace also can gather workload data for DEC DBMS databases and VAX ACMS applications.)

DEC Rdb has many predefined events that occur during run time. An event can have a start and an end, or it can simply occur. DECtrace allows events within DEC Rdb to be defined and data items to be associated with each event. These data items can be standard resource use items or items specific to applications using DEC Rdb.

DECtrace records several different pieces of information, called **items,** for each event. Items can be information about the event itself, such as the name of the event or the procedure where the event is occurring. Items can also include process statistics and performance information, such as the working set size at the time the event occurs.

For details on using DECtrace and RdbExpert with DEC Rdb applications, see the chapters in the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning* about using database statistics features to evaluate performance. For information about DECtrace, see the *DECtrace for OpenVMS User's Guide.* Consult the *DEC RdbExpert for VMS User's Guide* for details about RdbExpert.

8.6.2 Dependability Aspects of Application Form and Function

Users interact with the computing system in the course of using applications to accomplish their duties and, as a result, exert a major influence on the dependability of the system. How users enter application data should be a priority consideration in your list of error-reduction and fault-tolerant techniques. Planning application forms and functions with the intent of reducing the number of user errors requires a large commitment by developers and programmers during the application design stage. Because prudent application designers do not regard users as the default error-handling mechanisms, they plan to use automated systems operation tools during the application design.

Users assess dependability by how well the computing system helps them accomplish their work. Typically, users accomplish work through screen forms, command line interfaces, menu-driven interfaces, or others. Harmonizing data entry forms and interfaces with the user's work style is an important aspect of dependability. User forms and interfaces should be consistent so that operators are less apt to make input errors. If a data entry form is awkward or difficult to use, then the application is not completely dependable. This problem of enhancing system dependability through user interfaces is twofold:

- To implement a form or other user interface so that it is easy to use, the application software must minimize user frustration during data entry.

- To enhance user productivity with the application form or interface, express application training and documentation in a way that is familiar to the system users and customize the learning aids toward the particular needs of the users.

It is important that application forms and user interfaces are similar for all applications on the system. For example, if users are accustomed to overwriting form data to correct input errors, user interfaces should not implement a delete-and-enter method for users to correct their errors. You might also demotivate users if you introduce a new menu-driven interface to those who are accustomed to a command-line interface. Similarly, an inability to do an online retrieval for certain types of system messages severely degrades the productivity of the user who must instead manually reenter the information from hardcopy messages source code.

Because user errors typically involve the complexities of human personalities and behaviors, input errors can be difficult to predict and prevent. For applications and databases that spend a large amount of time sending and receiving information to and from terminal forms, you might consider using a forms product such as DECforms. DECforms lets you integrate text and simple graphics into forms. Application programs use these forms as user interfaces. DECforms is a Digital implementation of a proposed ANSI and ISO standard for a Form Interface Management System (FIMS).

Purchasing a forms interface product such as DECforms provides several advantages over programming your own menu or other user interfaces that can result in quicker, more accurate forms creation. The advantages include:

- Creating forms with a forms software tool results in simpler programming and more maintainable applications. Most forms products provide easy-to-use interfaces for designing and implementing a form with applications. The program is concerned only about the data to be displayed or collected. The program does not need to know where or in what format that data is placed on the screen.
- Detailed interaction with the user can be handled by the form. In most cases, the form can trap data entry errors before they are returned to the application or the database.
- A forms interface separates data entry from the functions the application must perform to process the data. The separation makes your application program device-independent, which allows easier distribution of your application. There is no down time due to reprogramming to accept new devices because the application remains unaware of the device with which it is dealing.
- Most forms interfaces allow you to quickly prototype an application to discover user requirements without having to code the entire application. The application will be available for use more quickly, and the ability of the user to test the forms early in the design process results in a form that is more suitable to user requirements.
- In some cases, a separate form interface provides the ability for front-end terminals in a VMScluster system to fail over automatically from one front-end processor to another. See ACMS Provides Automatic Front-End Terminal Failover for more information.

DECforms is particularly effective because it allows applications to support new devices without having to rewrite the application interface. For more information about DECforms, see the *DECforms Guide to Commands and Utilities*.

8.6.3 Dependability Characteristics of Transaction Processing Monitors

Because transaction processing (TP) applications are typically described as mission-critical or “bet your business” applications, they need the highest degree of availability, security, and integrity. The following sections describe the dependability features inherent in TP monitors and specifically in the VAX ACMS TP monitor.

8.6.3.1 How TP Monitors Can Assist Dependability Goals of Production Systems

A TP monitor is a collection of components bundled together in a tightly integrated package to facilitate application development and provide high performance and dependability in the production environment. TP monitors have inherent features that enhance performance and dependability, including any number of the features in the following list:

- Queue management
- Load balancing and scheduling
- Application scheduling
- Deferred transaction scheduling
- Transaction routing
- Administration utilities such as terminal server management and utilization management
- Security
- Menu subsystems
- Interfaces to nonstandard or other-vendor devices
- Message recovery
- Debugging support

The VAX ACMS product is a highly reliable TP system that includes these specialized features to keep applications up and running at optimal efficiency, recoverability, and durability. ACMS is also fully integrated with DEC Rdb, DEC DBMS, DECdtm services, DECforms, and the CDD/Repository. This group of products is part of the Digital COHESION environment for providing development, run-time, and management features for transaction processing applications. The high availability features inherent in these software products combine to provide smooth transitions when problems occur, even when handling a broad range of system failures.

ACMS takes full advantage of the DECdtm services provided with the VMS operating system to ensure consistent transaction commits, aborts, and recoveries even when failures occur. The DECdtm services use a two-phase commit transaction protocol to coordinate the flow of transaction requests and make sure that all the operations of a transaction are performed or none of them are performed. If all the operations are completed, then the transaction is committed and the database or databases are all in a consistent state.

It is important to note that ACMS and the DECdtm services deal only with application recovery, which includes recovery actions on or by the database, the ACMS TP monitor, the application program, and the application users. Thus, it is important to make sure that you also provide for system recovery. The VAX ACMS TP monitor, for example, supports the full range of VAX processors from VAXstations to the VAX 9000 and the fault-tolerant VAXft series of processors so that you can customize the configuration with full hardware redundancy, fault tolerance, or both to meet your needs.

8.6.3.2 How VAX ACMS Can Assist the Dependability of Production Systems

Combining VAX ACMS with DEC Rdb database management enhances your computing system's ability to resist most failures. For example, when the back end of an ACMS system fails, the ACMS TP monitor uses a search list to find and establish a link between the front-end processor and an alternate back-end processor where it can continue to perform online database processing. If you implement queuing on the ACMS system, transactions are queued in a queue repository until ACMS can process the transactions on the alternate back-end processor. The DEC Rdb database handles all aspects of database recovery.

The following sections discuss the main areas of VAX ACMS that provide fault management or recovery from failures that can occur during run time. The discussions provide a description of how each feature can be used and its implementation. See the VAX ACMS documentation set for more information.

ACMS Balances Process Pools

ACMS implements online and queued transaction requests using VMS processes. The ACMS TP monitor intelligently balances application user needs by automatically creating and removing processes as the work required of the system increases and decreases. Balancing process pools is an effective technique for providing increased application availability because when one process fails, there are other processes still available to provide service.

ACMS uses the minimum number of processes necessary to accomplish its work. As users sign in to ACMS, the TP monitor automatically balances the load by creating additional ACMS command processes to handle the additional terminals. Similarly, when enough users select a particular task so that the server processes that do the work for the task are overloaded, ACMS dynamically creates more server processes as necessary to avoid having to wait for tasks to be completed.

ACMS can also delete processes (from the process pools) when the work load is small. ACMS automatically maintains the pool even in the face of run-time errors that cause unnecessary process deletion. ACMS can dynamically regulate processes creation and deletion because it is capable of detecting the deletion of one of the pool processes and can take action to recreate another process if necessary.

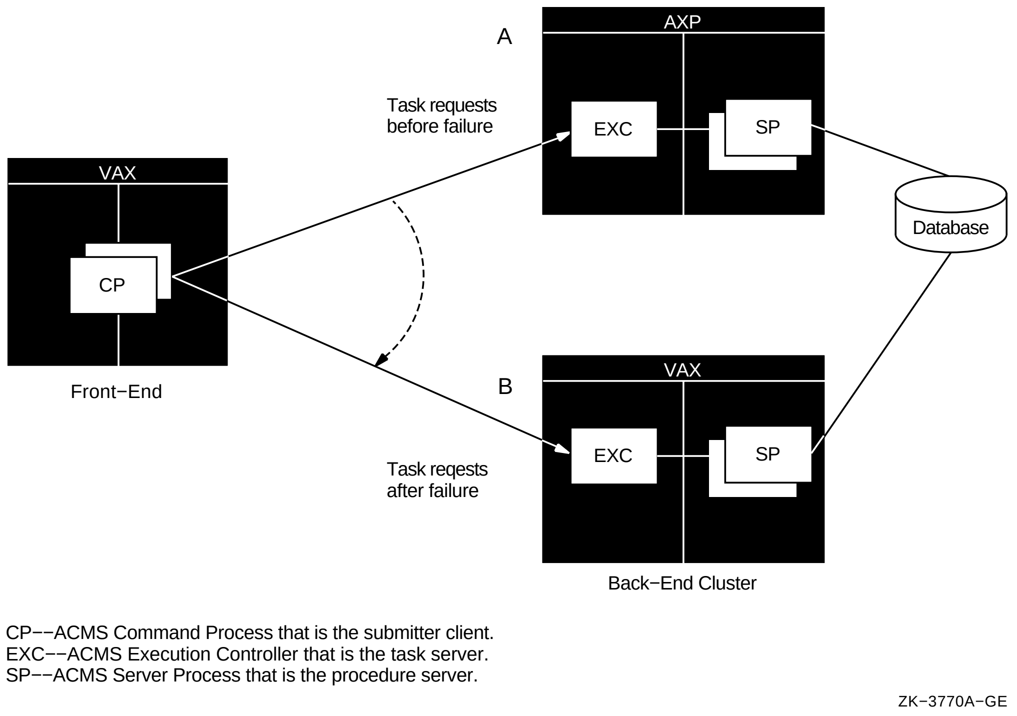
If the failed process is a permanent process, then the system might run without its full configuration of processes. You control the minimum and maximum number of processes in the process pools and the intervals within which the ACMS TP monitor creates or deletes the processes with the ACMS application definition language (ADU) and the ACMS management utilites. For complete details of how ACMS performs load balancing in process pools, consult *VAX ACMS Managing Applications*.

ACMS Fails Over Applications

ACMS automatically routes new task requests to a secondary application instance if the primary application instance becomes unavailable. The secondary application must be started on another node. When the application fails, an application node fails, or the link between the submitter node and the application node is severed for some reason, ACMS cancels all active tasks on the submitter node that are associated with the failed application, node, or link.

When a user selects a task in the failed application, ACMS attempts to locate a functioning instance of the application. Once an available node or application is found, all subsequent task selections are redirected to the available application and terminal users are once again able to select tasks. Figure 8–14 shows how ACMS application failover works.

Figure 8–14 ACMS Application Failover


AXP
A
Task requests before failure
EXC
SP
VAX
CP
Database
Front−End
VAX
B
EXC
SP
Task reqests after failure
Back−End Cluster
CP−−ACMS Command Process that is the submitter client.
EXC−−ACMS Execution Controller that is the task server.
SP−−ACMS Server Process that is the procedure server.
ZK−3770A−GE

The context of the application is lost during the failover to the new node. Thus, all work in progress at the time of the failure is canceled and the work must be resubmitted to the ACMS application. (See ACMS Uses Queues to Capture User Requests for information about saving work in progress at the time of a failure.) Terminal users receive an error message when the application failover situation occurs. Note that transactions that have already been committed to the database are not considered work in progress and as such as not affected by the application failure.

For this type of failover to work correctly, the back-end node that executes the multiple instances of the application must have access to the same database as the original node. Figure 8–14 illustrates how to configure the back end as a VMScluster system. When a node fails in a VMScluster system, DEC Rdb automatically rolls back unfinished transactions that were active on the failed node. The roll back occurs on another node in the VMScluster. Users cannot update the database until the database roll backs are complete. For more information on database roll back, consult the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning*.

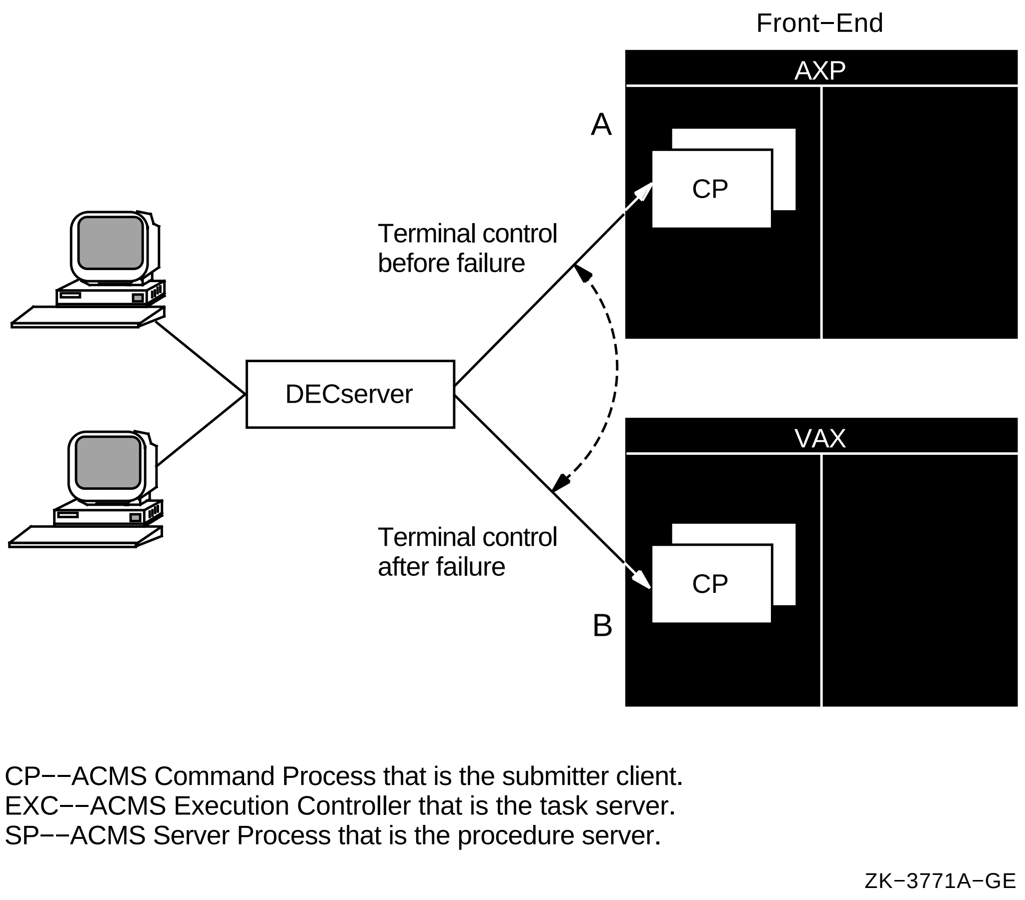
You achieve application failover using OpenVMS search lists and an ACMS service. You can define an application as a search list; that is, a list where each element in the list is another separate instance of the application. Thus, the application INVENTORY can be defined in the search list as the list A::INVENTORY, B::INVENTORY. The list indicates that the INVENTORY application can be found on nodes A and B. A search list can contain more than two elements. ACMS provides a service that you use to locate a functioning application instance in the VMScluster. The service acts as a name server and an application finder. (See the *VAX ACMS Systems Interface Programming* for information about the ACMS Systems Interface services.)

ACMS Provides Automatic Front-End Terminal Failover

ACMS provides the ability for a terminal in a VMScluster system to fail over automatically from one front end to another when the device is a LAT terminal that is controlled by ACMS. (See *VAX ACMS Managing Applications* for more information about ACMS-controlled LAT terminals.) When a front-end node fails, the user is unlikely to see any warning message at the time of the failure. All information and work on the terminal screen is lost. ACMS cancels tasks that are executing on the back end and rolls back uncommitted transactions. When the failover is complete, the user must sign in to ACMS again and reenter the last incomplete transaction.

Figure 8–15 shows how terminal failover works.

Figure 8–15 ACMS Front-End Terminal Failover


Front−End
AXP
A
CP
Terminal control before failure
DECserver
VAX
Terminal control after failure
CP
B
CP−−ACMS Command Process that is the submitter client.
EXC−−ACMS Execution Controller that is the task server.
SP−−ACMS Server Process that is the procedure server.
ZK−3771A−GE

When the ACMS system starts up, both front-end nodes try to control the terminals but only one of the front-end nodes can succeed. The other node requests control of the terminals so that when node A fails and loses the connection to the terminals, node B is automatically connected to and controls the terminals.

Implementing effective load balancing and terminal failover on the front-end machines can require extra work during configuration. For example, assume that you want to connect 100 terminals to two front-end nodes. For load balancing, connect 50 terminals to each front-end node. For availability, however, you must provide a way for the terminals to fail over to the other node. One way to achieve failover is to:

- Configure terminals 1–50 as ACMS LAT-controlled terminals on node A and terminals 51–100 on node B.
- Start the ACMS terminal subsystems on both front ends. The configuration will cause each front end to control half of the terminals.
- Add terminals 51–100 to the configuration file on node A and terminals 1–50 to node B.

- Issue the command ACMS/RESET terminals on each front-end node. This command causes ACMS to reread the configuration file. Each front-end node will continue to control its original terminals while also seeking to control the terminals on the other node.

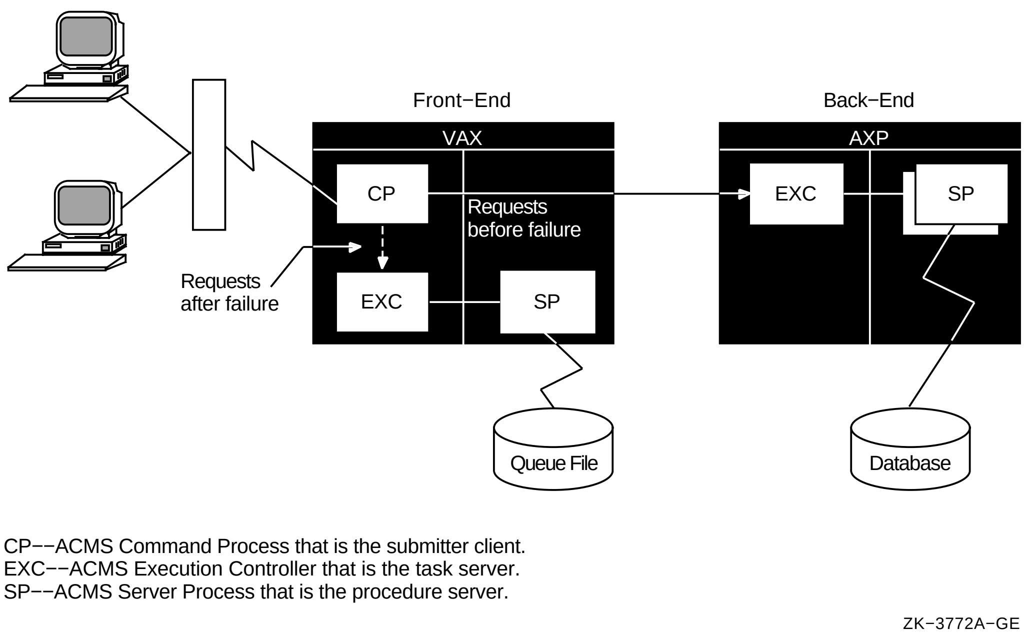
When a failover occurs, the terminals on the failing node automatically fail over to the remaining front-end node. Note that the performance of the surviving front-end might suffer when the failover occurs.

ACMS Uses Queues to Capture User Requests

ACMS queuing allows users to submit work even when an application is not available. When a failure occurs, it is possible to store user task requests in queues until the application or an alternate application becomes available. When the application becomes available, the stored requests are removed from the queue and the application executes the requests against the database.

Figure 8–16 shows a possible solution for capturing user requests in queues.

Figure 8–16 ACMS Request Capture



CP−−ACMS Command Process that is the submitter client.
EXC−−ACMS Execution Controller that is the task server.
SP−−ACMS Server Process that is the procedure server.

Normally, the application executes on the back-end processor and ACMS captures user requests on the front-end processor and routes them to the back end. When the back-end processor fails the application fails over to another application on the front-end processor, which captures user data and stores user requests in a queue repository. Note that there are two distinct ACMS applications in this solution: the application on the back end that ACMS uses under normal circumstances and the application on the front end that ACMS uses to capture requests and to queue them when the back-end processor is unavailable. ACMS switching between the two applications occurs automatically using the application failover mechanism described in ACMS Fails Over Applications .

You include ACMS queuing by including calls to ACMS queuing services in your application programs. See the *VAX ACMS Writing Applications* manual for information about programming requirements for ACMS queuing.

Another alternative is the DECtp Desktop for ACMS (Desktop ACMS) software product. Desktop ACMS lets you retain the robustness of ACMS applications on the back end while providing flexible and familiar graphical user interfaces from your own desktop programs. You can write your desktop programs to fail over to alternate ACMS nodes if the primary ACMS node fails. Refer to the *DECtp Desktop for ACMS Progamming Guide* manual for information about Desktop ACMS.

8.7 Managing Shared Information with Software Tools

You can enhance the dependability of your software when you facilitate central control of the development tools, program modules, and shared data for a distributed application development environment. You should create an environment that makes data available to all users that need to share that information. The following sections explain how to create and manage shared data using the Digital COHESION environment, OpenVMS logical names, and DCL command procedures.

8.7.1 Using the Digital COHESION Environment

The Digital COHESION environment addresses the need to develop and manage software and data that will run on a variety of platforms, and create applications with functions that can be distributed across VMScluster systems and multiple-vendor platforms. By providing a shared development environment, COHESION provides high levels of system availability, security, and data integrity in computing systems.

The COHESION environment provides several DECset™ software tools to help you share and manage application objects and data on the VMS operating system, including:

- DEC Language-Sensitive Editor/Source Code Analyzer (DEC LSE/SCA)

  Use DEC LSE/SCA to develop and debug source code modules for applications. DEC LSE/SCA provides a development environment that includes templates and online help about specific commands or clauses. DEC LSE/SCA assists you in creating and navigating in program source code and modules by providing special LSE key definitions to get language-specific help and to fill in incomplete keywords and structures. See the *VAX Language-Sensitive Editor and VAX Source Code Analyzer User Manual* documentation for more information.

- DEC Code Management System

  DEC CMS builds dependability into your software development environment by tracking everything that happens to project files during development. DEC CMS is an automated file librarian that tracks revisions of source code, documentation, data files, test files, system build descriptions, and requirements documents. As a result of DEC CMS recording every change when it was made and by whom, developers are always working with the most up-to-date files and can reconstruct any phase in the code development process. The more complicated the project, the more important managing the application source code becomes. See the *Guide to VAX DEC/Code Management System* documentation for more information.

- DEC Test Manager

DEC Test Manager is a tool for creating, organizing, and performing tests of software applications. There are three main components to a DEC Test Manager test system:

  - The DEC Test Manager library stores information about tests.
  - The DEC Test Manager test descriptions describes an individual test and how it is run.
  - The DEC Test Manager collection identifies a collection of tests to be run as a group.

  The main repository for information about DEC Test Manager is the DEC Test Manager library. It is much like the DEC CMS library and is created in the same manner. See the *Guide to VAX DEC/Test Manager* documentation for more information.

- DEC Module Management System

  DEC MMS makes applications available quickly by automatically rebuilding the system after programmers make changes to application code. DEC MMS saves significant processing time because it has the intelligence to recompile and to link only the modules that have changed since the last system build. See the *Guide to VAX DEC/Module Management System* documentation for more information.

- DEC Performance and Coverage Analyzer

  Unfortunately, even if you design the application correctly, it might not always run as expected. The major areas you need to look at include the user interface, the application source code, the system components, and the database. You can use DEC PCA to identify possible performance problems in your application. DEC PCA is also useful for identifying and checking logic in the code that might not have been executed during testing. See the *Guide to VAX Performance and Coverage Analyzer* documentation for more information.

The DECset products manage centralized or distributed application data through the CDD/Repository. By using the CDD/Repository, you provide a single point of control for defining and sharing access to software development information and gain several advantages in regards to data availability. By defining data once in the repository, you improve the availability of the data by:

- Avoiding the redundancy of having to define the data in every application. Without a repository, you have to define the data in every piece of an application, leading to redundacy of and possible inconsistency in data definitions. With the CDD/Repository, you can copy the definition from the repository into every piece of the application that uses the data. If the data changes, change only one view of the definition in the repository and rebuild the application. The change is automatically reflected and the application is quickly available for use.
- Creating and compiling the actual data values available in any of several ways: for example, with a database management system such as VAX ACMS, DEC Rdb (with SQL) or DEC DBMS, RMS files, DECdesign, DEC RALLY, VIDA with DB2, and integration with most of the Digital languages and several other-vendor languages. When you create software information, it can be automatically stored in the repository. For example, when the database designer compiles an SQL schema definition that specifies a repository path name, DEC Rdb inserts record definitions in the CDD/Repository.

When application source code needs to specify parameters that correspond to database tables, the code can then refer to matching records in the CDD/Repository.

- Ensuring the repository is accessible across a VMScluster. The definitions in the repository are highly available because you can store the dictionary redundantly on several nodes in a cluster. The CDD/Repository maintains consistency across the dictionaries on the VMScluster.

You do not need to know how the software information is stored in the CDD/Repository. However, you need to know where the information is stored and how to access it from high-level language programs. For more information on using the CDD/Repository, see the *Using CDD/Repository on VMS Systems* manual.

8.7.2 Defining Symbols and Logical Names

In addition to using the COHESION environment to provide a common view of source files and data definitions, you need to effectively manage several OpenVMS directories for other shared information. Application availability depends on how effectively programmers use common symbols and logical names to locate application modules. It is best to use logical names rather than specific OpenVMS device and directory names, both in the development environment and more importantly in the application itself. For instance, if all programmers use their own directories to store object modules, it is extremely difficult to find the latest versions when it comes time to link the entire application.

Logical names are easier to remember and can identify the type of files the directories contain. As an example, imagine how programmers could glance at the logical name PAYROLL_OBJ to know that the directory contains .OBJ files for the payroll application or that the PAYROLL_SRC directory contains the source files. In addition, logical names that include a disk specification in addition to the directory path can prevent later problems when you move applications from the development environment to the production environment.

Thus, when the application moves to a new disk or to another VMS system, all you need to do is change the definitions for the logical names. No changes need to be made to the application source code or to any command procedures that build the application. Using logical names can save many last-minute changes and can avoid the resulting delays in application dependability on the production system for which it is designed.

You define logical names using the DCL command DEFINE. To make sure everyone working on the application uses the same definitions, write a single command procedure containing the logical name definitions and invoke it from within either individual LOGIN.COM files or the system login command file, SYS$MANAGER:SYLOGIN. See the *OpenVMS System Manager's Manual: Essentials* for more information.

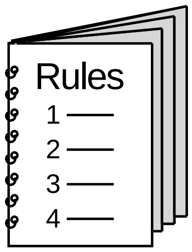
8.7.3 Using the DNS Namespace

VAX Distributed Name Service (DNS) is a software product that provides a network-wide registry of shareable resources. DNS keeps track of the location of file resources to which the Digital Distributed File Service (DFS) manages access. Using DNS allows users on one node to access files or libraries on a remote node. DNS manages this access without the users being aware of the location of the files. Users access the remote files in the same manner that they access local files.

In a large development environment where many programmers must share source files or data, DNS allows each programmer to check out a particular source file, edit it on his or her own system, and return it. When the programmer also uses VAX DEC/CMS to maintain a history of changes to the source files and all programmers work from the same VAX DEC/CMS libraries, the code is centralized such that it remains on one system in the network.

See Section 7.7 for summary information about how DNS can help you establish control over the naming of your shared nework resources. See the *VAX Distributed Name Service Management* manual for details about DNS.

9 Dependable Data Center Techniques



The need for efficient and controlled system and data center management has become more urgent as computing systems grow in physical size, capacity, and complexity. In the beginning, computer operations were typically managed by full-time computer operators behind the glass walls of an air-conditioned, raised-floor data center. Security was often a matter of putting a lock on the door to restrict access to the data center. With the advent of minicomputers, personal computers, and desktop workstations, computers are being located wherever they are needed. Network and system managers are responsible for a dynamic, unpredictable environment consisting of large numbers of diverse systems and used by large numbers of people performing varied tasks. Centralized data centers are being augmented and sometimes replaced with a distributed environment that can include a number of network technologies and in many cases multiple vendors, operating systems, or both. The geographically and organizationally distributed locations of the equipment complicate all aspects of data center operations and system management, especially the area of system security.

9.1 Managing Complex Computing Environments

Managing complex computing environments often involves the use of multiple products and services. Table 9–1 relates a wide range of system management tasks to the tools and utilities that can help you perform the tasks.

Table 9–1 Data Center Management Portfolio

Software Products and Features	Dependability Impact
Accounting	
OpenVMS ACCOUNTING	Facilitates user control
POLYCENTER Performance Solution	Provides basic user accounting and chargeback reports
Application Management	
ALL–IN–1™	Controls user activity
VAX ACMS and DECtp Desktop for ACMS	Provide recoverable applications using a transaction processing (TP) monitor
DECdecision	Offers a method of collecting and categorizing data
DEC RALLY	Provides user-friendly interface for TP
VAX TEAMDATA	Allows data collection and analysis
LMCP (Log Management Control Program)	Maintains transaction logs
Application and System Debugging	
VMS Debugger	Facilitates application reliability
DEC Performance and Coverage Analyzer (DEC PCA)	Tunes and checks applications
ANALYZE/AUDIT command	Invokes the Audit Analysis Utility to selectively extract information from the system security audit log file
ANALYZE/CRASH_DUMP command	Enables Digital service personnel to determine cause of failure
ANALYZE/DISK_STRUCTURE command	Checks and repairs on-disk file system
ANALYZE/ERROR_LOG command	Formats log of device history (aid during servicing)
ANALYZE/IMAGE command	Enables programmer to examine inner workings of executable images
ANALYZE/MEDIA command	Checks non-MSCP disks, diskettes, and tape cartridges for bad blocks
ANALYZE/OBJECT command	Enables programmer to examine object modules
ANALYZE/PROCESS_DUMP command	Facilitates application reliability
ANALYZE/RMS_FILE command	Facilitates file I/O tuning
ANALYZE/SYSTEM command	Enables the system manager to examine inner workings of the running system
Archiving	
VAX Storage Library System (SLS)	Protects user data on other media

(continued on next page)

Table 9–1 (Cont.) Data Center Management Portfolio

Software Products and Features	Dependability Impact
	Data Caching
RMS caching (local and global buffers)	Alleviates I/O bottlenecks
	Data in Memory
DECram (in-memory disk/device)	Alleviates I/O bottlenecks
	Interactive Services
Command procedures or F$LEXICAL functions	Automate routine system management tasks
DIGITAL Command Language (DCL)	Provides a consistent user interface
DCL Command Procedures	Provide a full programming language for operator and user interfaces
DEC TPU, EVE, and other editors	Include editing interface with full session journaling
OpenVMS Mail Utility	Offers notification services using a callable interface
	Capacity Planning
VAX Performance Advisor	Prevents bottlenecks proactively
POLYCENTER Performance Solution	Provides a capacity planning function to help the system manager analyze how changes in the configuration affect users performance
	Event Messages
OPCOM message routing	Provides event notification
VMScluster Console System (VCS)	Consolidates system consoles and analyzes console messages for events
DECalert	Manages sophisticated event notification; please see the note that appears at the end of this table
Data Center Management (DCM)	Detects and reports events on running OpenVMS systems polled using DECnet

(continued on next page)

Table 9–1 (Cont.) Data Center Management Portfolio

Software Products and Features	Dependability Impact
	Data Access
DEC DBMS	Offers a CODASYL-compliant database with journaling, online backup, online database reconfiguration, and TP support
VAX RMS™	Provides a highly tuneable file access with write-through caching
RMS Journaling	Provides data recovery and TP support
DEC Rdb	Offers a relational database with journaling, online backup, selected online database reconfiguration, and TP support
	File Operations
BACKUP Utility	Saves and restores user data on other media
CONVERT Utility	Maintains optimal ISAM file performance
COPY command	Moves files around for I/O tuning
DELETE/ERASE command	Provides extra privacy
DUMP command	Displays actual file contents to facilitate application debugging
PURGE command	Facilitates disk space reclamation
RESTORE command	Applies RMS journals to recover RMS files
SET FILE command	Manipulates file characters
TYPE command	Displays batch log files during their execution
	Disk Striping
VAX Disk Striping Driver	Spreads I/O load over multiple disk drives
	Encryption
VAX Encryption	Provides added security/privacy with data encryption services
DESNC LAN Encryption Device	Provides D.E.S. (data encryption services) encrypted Ethernet messages
	Forms Management
DECforms	Separates form input from application management
	Job Scheduling
DECscheduler	Schedules batch and print jobs
Generic queues	Uses VMScluster queues to feed node-specific queues across the VMScluster
Execution queues (batch and print)	See the Queue Management category

(continued on next page)

Table 9–1 (Cont.) Data Center Management Portfolio

Software Products and Features	Dependability Impact
Use Up to 6 Processors per System Kernel	
Symmetrical multiprocessing (SMP)	Helps system survive CPU failure
Mirrored Disks	
Volume Shadowing for OpenVMS	Helps system survive disk failure
Disk Defragmentation	
DEC File Optimizer for OpenVMS	Defragments disks by using the OpenVMS **movefile** subfunction
Multiple-Vendor File Operations	
KERMIT	Facilitates personal computer file transfer
PATHWORKS	Provides personal computer integration (OS/2®, MS–DOS®, Macintosh)
ULTRIX™ Connection Product	Allows NFS access from UNIX® (TCP/IP)
VAXlink	Allows VSAM and IMS data access
VIDA	Provides DB2 real-time access
SNA Gateway access routines	Facilitates LUG.2 communication with IBM® programs
Online System Configuration	
SYSGEN (System Generation utility)	On VAX systems, configures all connected devices on a system boot
SYSMAN (System Management utility)	On AXP systems, configures all connected devices on a system boot; also enables device and processor control commands to take effect across a VMScluster
Remote Bridge Management System (RBMS)	Maintains a database of the setup of a system's complement of network servers, and facilitates automatic loading of the servers upon system initialization
Terminal Server Manager (TSM)	Maintains a database of the setup of a system's complement of terminal servers, and facilitates automatic loading of the servers upon system initialization
VAXsimPLUS	Monitors device status and alerts the system manager of impending failure

(continued on next page)

Table 9–1 (Cont.) Data Center Management Portfolio

Software Products and Features	Dependability Impact
Performance Management	
DECamds	Precisely identifies, in realtime, causes of problems in computing components on the local area network (LAN).
AUTOGEN command procedure	Optimizes system parameter settings based on usage
POLYCENTER Performance Solution	Historical data that captures bottlenecks over a user-specified period of time and recommends ways to fix problems
VAX Performance Advisor	Provides detailed performance data and graphs
VAX Software Performance Monitor (SPM)	Provides detailed performance data and graphs
OpenVMS Monitor Utility	Provides basic performance data
Queue Management	
DELETE/QUEUE command	Deletes a print or batch queue and all the jobs in the queue
INITIALIZE/QUEUE command	Creates or initializes queus and assigns names and attributes to the queues
START/QUEUE command	Starts or restarts the specified queue after it has been initialized
STOP/QUEUE command	Pauses the specified queue and suspends all the jobs currently executing on the queue
SET QUEUE command	Changes the attributes of the queue
SHOW QUEUE command	Displays information about queues and the jobs currently in queues
SHOW ENTRY command	Displays information about a user's batch and print jobs or about specific job entries
SUBMIT command	Queues one or more batch jobs to a batch queue
PRINT command	Queues one or more files for printing to an output queue
Remote System Management	
DECmcc	Performs industry standard network and system management system functions
VAXrsm (Remote System Manager)	Provides remote management of OpenVMS or ULTRIX nodes
SYSMAN Utility	Performs system management commands across nodes in a VMScluster

(continued on next page)

Table 9–1 (Cont.) Data Center Management Portfolio

Software Products and Features	Dependability Impact
Security	
AUTHORIZE (Authorize utility)	Modifies user profiles
OpenVMS C2 features	OpenVMS on AXP and VAX systems provides C2 level of security, as defined by the U.S. government. The C2 features provide object protection and auditing.
OpenVMS Security Enhancement Service (B1 services)	Offers consulting and software to implement “B1” level of security
SET ACL command for Access Control Lists (ACLs)	Sets complex protection on many system objects
SET AUDIT command for security auditing	Facilitates tracking of sensitive system objects
SET SECURITY command	[tbs]
SHOW SECURITY command	[tbs]
Shared Disk Storage	
	See the Data Access category in this table for record level locking
Software Installation	
POLYCENTER Software Installation utility	Provides features that let you perform rapid installations and deinstallations, and manage information about already installed products.
VMSINSTAL	Provides semi-automatic software installation
LMF utility	Manages software licenses
System Configuration	
INITIALIZE command	Creates an on-media file system
MOUNT command	Makes a volume known to the system
SET DEVICE command	Manipulates device characteristics
SET CPU command	Starts or stops CPUs in an SMP kernel
SET MAGTAPE command	Manipulates magnetic tape characteristics
SET PRINTER command	Manipulates printer characteristics
Remote Bridge Management System (RBMS)	Maintains a database of the setup of a system’s complement of network servers, and facilitates automatic loading of the servers upon system initialization
Terminal Server Manager (TSM)	Maintains a database of the setup of a system’s complement of terminal servers, and facilitates automatic loading of the servers upon system initialization

Note

In February 1994, Digital granted Target Systems Corporation a nonexclusive, nontransferable license to use DECalert Version 1.2 software. Under the agreement, Target Systems may provide support for the Digital customer base (existing and future) for the product, as well as future development and migration of the product or its derivatives. Target Systems named the product TARGET–>ALERT.™ Digital reserves the right to continue to sell DECalert or its enhancements, as Digital deems fit.

System management and operational complexity alone can become a significant source of unavailability. Add to this the need to provide better performance and overall dependability and you quickly realize the huge responsibility involved in data center operations.

9.2 Data Center Operations

Many data center operations are the same whether the environment is:

- Small with one or two CPUs
- Large with several nodes
- Distributed VMScluster systems
- Networked

Common data center operations include:

- Managing the diverse activities of hardware and software installations
- Setting up and authorizing user accounts
- Monitoring system behavior and performance
- Monitoring batch and print queues
- Performing backups
- Scheduling and performing preventative maintenance tasks
- Attending to physical operational needs such as:
  - Mounting tapes
  - Servicing printers
  - Responding to user requests

Some of these operations represent work that you can schedule during times when system outages are allowable; that is, when the system components necessary to accomplish the work of the business do not need to be 100% available.

Contrast scheduled outages with unscheduled system outages such as sudden power outages, natural disasters, and sudden hardware failures. These outages can be devastating because they typically occur at a time and for a length of time that impacts business. Scheduled outages, while less disastrous, can still become significant when they become commonplace.

Ideally, your goal should be to perform data center operations in an unattended, self-managing data center environment. However, the reality is that most data center operations are done manually. In fact, for highly dependable systems, the additional availability you gain through redundant components (for example: volume shadow sets, additional CPUs, redundant networking) can sometimes require additional system management.

The following sections point out techniques that reduce failures related to system management activities and prevent them from rendering the system unavailable. In addition, you can refer to the following Digital documentation for more detailed assistance with system management tasks:

- *VMScluster Systems for OpenVMS*
- *OpenVMS System Manager's Manual: Essentials* and *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems*
- The *Guide to Data Center Management*

9.2.1 Using DCL Procedures to Minimize User Error

Data center personnel typically devote a lot of time to manually running production jobs. This is not only expensive in terms of salary and time, but it is also prone to errors and consequently, system unavailability. It is possible to reduce errors, to reduce data center costs, and to increase productivity and availability by automating the execution of repetitive jobs. You can automate the execution of repetitive production tasks by writing DCL command procedures or using a production scheduler. For more information about writing DCL command procedures, see the *OpenVMS System Manager's Manual* and the *OpenVMS DCL Dictionary*.

Production schedulers usually include the ability to schedule time and job dependencies, VMScluster system load balancing, and a windowed user interface for real-time job monitoring and control. Setting up production schedules with one of these tools can take significant planning but the benefits usually outweigh the investment effort. Digital offers DECscheduler for production scheduling. See the DECscheduler documentation for more information.

9.2.2 Scheduling Preventative Maintenance

For many components, you can lessen the frequency of failures that impact dependability by a regular program of preventative maintenance and symptom directed diagnosis. By scheduling time for the tasks such as the ones suggested in the following list, you can allow regular maintenance chores to take place without interrupting applications:

- Hardware—repairs, cleaning, adjustments, upgrades, installations. Hardware failures repaired or prevented through this approach can thereby extend the apparent reliability of the hardware.
- Software—upgrades, maintenance releases, patch installations, disk defragmentation, cleaning up error logs, verifying status and operation of fault management tools, and collecting data to analyze your needs for additional maintenance might also be done during this preventative maintenance time.

Maintenance tasks such as capacity and performance planning, configuration management, security evaluation, storage management, system tuning, data backup, and archiving often need some access to computer resources. By scheduling maintenance servicing for times when the business demands on the computing system are reduced, you can minimize or eliminate potential impact on dependability. Scheduled maintenance includes not only deferring remedial maintenance to an appointed time but also planning for hardware and software upgrades and value-added services.

For businesses with 24x365 requirements, a combination of fault tolerance and redundancy, failover capability, and online repair capability might be the best approach for continuous operation. For businesses requiring high availability during an 8-hour shift using fault tolerant hardware and avoiding common faults by deferring service coverage and maintenance to off-hours works best.

Configuring redundant components not only provides failover capability, but also allows some preventative maintenance (such as tape head cleaning) to occur on some devices while others remain in use.

9.3 Failures and Recovery

Hardware failures differ from software failures in that hardware failures are guaranteed to occur eventually. In the hardware world, mechanical devices eventually break down. The impact of hardware or software failures on system dependability is obvious and immediate. When someone says the computer is broken, this generally means the hardware is inoperative. Even when you have installed the best hardware available, failures can arise as the hardware ages.

Although more than one hardware component might fail at the same time, hardware failures usually occur independently of one another. A failure can occur when something wears out, is abused, suffers from design flaws, or is subjected to unusual environmental conditions such as high temperatures, humidity, or power failures. In addition, a severe software error could lead to an OpenVMS fatal bugcheck error and a computer shutdown.

A node failure might be detected in any of the following ways:

- By terminal users when normal processing stops
- By operators through data center monitoring products and alerting technologies such as DECalert
- In a VMScluster environment, where the VMScluster software can detect that a machine failed
- In a nonclustered environment, where DECnet learns of the node failure
- When terminal servers try to connect to a service and fail
- When a front-end node that has outstanding connections with a back-end application is notified (usually by means of a network connection error) of a back-end node failure

The typical data center response to node failure is to perform the following tasks:

- Reboot OpenVMS
- Start DECnet
- Mount user disks
- Install images
- Start batch and printer queues
- Start layered products
- Define systemwide logical names
- Start applications

For serious problems that require repair, the cause of the failure determines the type of repair that is necessary and the amount of time it takes to accomplish that repair:

- Hardware failures might require a call to Digital service personnel to replace a hardware module.
- Environmental failures require that you restore the computing environment to its proper specification.
- Software failures or system configuration errors might require disabling offending software, resetting parameters, and reconfiguring the system.

Not all hardware failures result in the system being unavailable. Failures in a single component might not have any impact on your business if the failure is made invisible (through either high-level fault management or hardware redundancy) or if the failure occurs at a time when the component is not critical to your business application.

For example, if a system that does not use Volume Shadowing for OpenVMS has a disk device that fails (thus rendering all media on the device unreadable and unusable), OpenVMS pauses all processes that are attempting to access the device while it attempts mount verification. If the device does not come back on line within a specified time (the amount of time that is specified by the system parameter MVTIMEOUT), then the OPCOM facility sends mount verification messages. These messages state that the disk is off line to the operator terminals and all the queued I/O requests complete with errors.

Note

Avoid running the OPCOM facility on every node in the cluster to reduce the amount of data transfer across the network. Be selective about the nodes on which OPCOM runs. However, monitor the facility to ensure that it runs on at least one node.

By following a regular backup procedure, using redundancy, and scheduling preventative maintenance with your service vendor, you can actively safeguard your system before problems cause an operational failure. In addition, by using products like Data Center Monitor (DCM), DECalert, VMScluster Console System (VCS), VAXsimPLUS, and the PASS service, you can help alert an operator of device failure. See the product-specific documentation on these products for more information.

Note

Do not hard wire operator consoles to VCS. If the console connection is broken, you also lose the connection to VCS. It is better to create a virtual terminal on which you can connect any VCS.

9.3.1 Catastrophic Failures

Catastrophic failures are usually easy to find and to repair. Examples of this type of failure include power supply burnout, disk head crashes, severed cables, and nonfunctioning logic on a circuit board. A severe software error could lead to an OpenVMS fatal bugcheck and a computer shutdown. The best way to prevent these types of failures from impacting the system users and applications is to build redundancy into your system at key points. Then, when a component or subsystem fails catastrophically, the surviving hardware can continue to provide service.

Pay careful attention to the actual effects of catastrophic failures even if the system remains operational. For instance, if the OpenVMS operating system detects a failed section of main memory in a user's process, it will prevent that memory from being used again until the problem is fixed. However, the particular user process that occupied the bad memory has to be deleted, even though the other users may not be impacted. The Digital VAXft systems are not vulnerable to this type of situation because the fault tolerant systems use fully redundant memory subsystems.

9.3.2 Intermittent Failures

Intermittent failures are usually more irritating than catastrophic failures because they play hide-and-seek with the service personnel. Examples of intermittent failures include the ability to read some magnetic tapes but not others, spots on disk media that require many retries to retrieve the data, and video displays that lose synchronization until you tap the side of the display monitor.

You can minimize the impact of intermittent failures on your system. The VAXsimPLUS product, for example, monitors soft or correctable error rates. When the error rate crosses a defined threshold, VAXsimPLUS notifies you to initiate corrective action before the failures become catastrophic. See the VAXsimPLUS documentation set for more information about capabilities of the product.

The most effective way you can help your service vendor isolate and fix intermittent failures is to have your data center personnel keep comprehensive written records on the conditions and actions surrounding the failures. Being able to duplicate the problem is the first step to diagnosing its cause. The necessary conditions for failure might include ambient temperature, electrostatic buildup, vibration from an external source, certain sequences of system, and actions by personnel.

9.3.3 Multiple-Cause Failures

Multiple-cause failures are the toughest problems to find because replacing a suspected component does not cure the problem. It merely changes the symptoms. Some examples of multiple-cause failures can include a faulty communications controller with a bad cable, a bad tape cartridge inserted into a faulty tape drive, or failed board-level voltage regulators plus an erratic power supply.

Sometimes, such as in the case of voltage regulators, the multiple causes of the failures are related. For example, an erratic power supply may cause the board-level voltage regulators to fail. Similarly, if the parameters of the main power source exceed the boundaries described in the hardware specification, the real cause could have been poor utility power.

A classic example of failures causing other failures occurs when you receive fatal disk errors and respond inappropriately. When you attempt to logically mount a removable disk pack that has been inserted in a disk drive and receive fatal disk error messages, the natural response is to physically move the disk pack to another drive and issue the MOUNT command for the new drive. Unfortunately, the act of moving a bad disk pack to a good drive could very likely cause the disk heads on the good drive to be damaged. Thus, you now have two disk drives that in turn could likely damage any good disk packs that are subsequently mounted on either disk drive. Obviously, this scenario can quickly produce many bad disk packs and damaged disk drives.

9.3.4 False Failures

Some hardware failures are actually caused by software defects or operator mistakes. Examples of false failures might include specifying the wrong label when attempting to mount a magnetic tape or missing the transmission of a message because insufficient buffering was allocated.

The best procedure for detecting and correcting this class of errors is, again, comprehensive recording of behavior and conditions. The documentation and practical knowledge that you gain from recording these events can help your service vendor redirect the search for the failure as quickly as possible.

Incompatibilities between hardware and software components can also cause intermittent faults that are difficult to diagnose. For example, as system configurations are becoming more open and hardware from multiple vendors is mixed, it is more difficult to track incompatibilities. Thus, it is essential that you take great care during the installation of new hardware and software to avoid all known incompatibilities.

9.4 Upgrades and Installations

The ease and accuracy of installing hardware, hardware options, or software can significantly impact the dependability of both the operating system and applications. Installations can take anywhere from minutes to days. The installation of additional equipment can require loss of system usage for a period of time. As you plan for a product upgrade or installation, keep in mind the following advice:

- Test the software installations on a separate test system before installing the software on the production system. Perform load tests for some time using realistic data. See Section 9.4.1.
- Preplan the installation to avoid having to back out of a partial installation. Backing out of a partial installation can be difficult and lead to lengthy down time.
- Ensure compatibility with other hardware and software on the system by performing upgrades to separate units, changing application or operating system parameters, or altering operating procedures. See also Section 9.4.2.
- Verify a product's installation by running the verification procedure to ensure that the installation was successful.

9.4.1 Continuing Service to Users During Upgrades and Installations

There is less disruption in service if, when you do system upgrades and installations, you thoroughly test the new software and integrate it with layered or other-vendor products on a nonproduction system first. The benefits outweigh the disadvantages of temporarily supporting two software versions. Although you must use the older version of the software for production work while the new version is being tested, this method allows you to find and to fix product or program incompatibilities without loss of production system availability.

9.4.2 Controlling Quotas and Privileges

To add applications and users and to configure OpenVMS for use, you must also adjust numerous system parameters. The minimum value for a parameter may be sufficient for the installation or operation of an application, but may not be the correct value for optimum use or performance. When critical parameters are set to inappropriate values, the result can be system or application crashes. Determine the appropriate parameter values by:

- Reading the operating system and applications documentation
- Running AUTOGEN periodically
- Using POLYCENTER Performance Solution
- Consulting with your service vendors

Because applications software introduces additional complexity, you must adjust system parameters and quotas just to perform the installation as well as to address interaction and compatibility issues.

Follow the guidelines and adjust the parameters and quotas as recommended in the product’s release notes and installation guides. Because of the complexity that additional applications software introduces, consult with your software specialist to help you adjust system parameters and quotas and address different software interaction and compatibility issues.

Because the installation of one software product can interfere with other previously installed products or can require those products be upgraded, you must reevaluate parameter settings each time you add to or make a change to the software running on a system.

9.5 Backup Procedures

Backing up file and system data reduces the availability of your system and data because you may need to shut down some applications to perform the backup. However, as part of the regular data center procedures, operators should copy operating system files, application software files, and associated files to an alternate device using the OpenVMS Backup utility (BACKUP).

Note

Back up DEC DBMS and DEC Rdb database files with the DBO/BACKUP and RMU/BACKUP commands, respectively. Do not use BACKUP with DEC DBMS and DEC Rdb database files. See Section 8.6.1.2, the *VAX Rdb/VMS Guide to Database Maintenance*, and the *VAX Rdb/VMS Guide to Database Performance and Tuning* for more information.

Backups are not the most exciting data center job, but they are an essential first step toward recovering from a future disk or node failure, or from user error that results in the deletion of essential files. It is important to develop and implement backup procedures that include:

- Backup process for the operating system or systems, including a schedule of the backup frequency and a location for the backup media.
- Backup process for data files, including a schedule of backup frequency and a location for the backup media.
- Backup process for application software, including a schedule of backup frequency and a location for the backup media.
- Off-site storage for:
  - A backup copy of the operating system or systems
  - Duplicate computer programs
  - Duplicate software and program documentation
  - Duplicate data files

  Software documentation should be available off site to describe the recovery purposes.
- Plans for regularly scheduled:
  - Daily incremental backups
  - Weekly full backups
  - Full backups for off-site storage

The backup processes must be consistent with application and user needs. This may require creative scheduling so that you can coordinate backups with times when user and application system requirements are low.

Ideally, the backup procedure should copy the data to a large disk that has a large amount of contiguous empty disk space. It is to your advantage to check the backup disk or to write a DCL command procedure that can automatically spin through your backup disks to see which disks have enough room to store the backup data.

Because the backup process might involve shutting down some applications, you must be able to schedule backups for times when system unavailability will not cripple the business. If you cannot afford to shut down the system regularly for the time it takes to do a full backup, it is possible to use shadowed disks to obtain backup copies of data with only brief interruptions of system and data availability. Because shadowed disks are identical copies of each other, you can use one of the shadow set members as a backup disk. (Note that using shadow set members cannot protect against accidental deletion of files.)

You can add a disk to an existing shadow set to produce a copy of the data on the shadow set. Once the copy operation completes, dismount the entire shadow set; at this point, each disk is identical to the other disks of the former shadow set. Even though this means taking the shadow set off line temporarily, dismounting the shadow set is the recommended method for producing an identical copy on a backup disk.

See the *Volume Shadowing for OpenVMS* manual for more information about using volume shadowing to perform backup operations on a shadow set.

Note

Do not dismount a member of a shadow set to back up a DEC DBMS or a DEC Rdb database. Instead use the DBO/BACKUP or RMU/BACKUP commands. See Section 8.6.1.2 for more information.

Archiving backup data can be accomplished by executing archiving procedures written inhouse or by using the VAX Storage Library System (SLS). SLS is an OpenVMS layered software product that provides the ability to manage data archiving and OpenVMS backup activities. SLS manages the resulting media, the media's location, and the devices associated with the archive activities and backup activities. Using SLS is advantageous from a dependability standpoint because you can quickly retrieve all information about backups or archived media. See the SLS documentation for details.

Finally, a good backup procedure is useless unless you are able to quickly locate and restore the lost data. Before you have an equipment failure, test and evaluate your data center personnel's ability to restart the system and recover data from backup media to an alternate disk volume.

9.6 Dependable Disk Devices

When a device fails and is repaired, it can require additional effort to make it available for use. Disks in particular require initialization and restoration of backup data. The disks are not available during restore and recovery operations.

Another problem with disk dependability is **disk fragmentation**, where data file extents are so scattered across a disk that I/O access performance diminishes to the point where the disk might appear to be unavailable.

9.6.1 Restoring Disk Devices Containing Databases

Recovery from a failure of a disk that contains DEC Rdb database files must begin before the failure occurs. As part of regular database maintenance, the operator copies the database to an alternate media with the RMU/BACKUP command. After the failure occurs, the database or certain areas of the database are restored to the replacement device using the RMU/RESTORE command. This returns the database to a consistent, preexisting condition. The database is rolled forward to the current state by applying the changes stored in the after-image journal (AIJ). The database AIJ file is created during normal processing and contains a complete record of all the changes that have been made to the database since the last RMU/BACKUP operation. The RMU/RECOVER command is used to perform the roll forward operation.

See the *VAX Rdb/VMS Guide to Database Maintenance* and *VAX Rdb/VMS Guide to Database Performance and Tuning* for details about database backup and recovery operations.

9.6.2 Defragmenting Disks to Improve I/O Performance

Over time, the process of adding, modifying, extending, and deleting files on a disk device can lead to file fragmentation. Fragmented files may hinder overall I/O performance because the applications must wait while the pieces of fragmented files are located across the disk and brought into memory. A full OpenVMS image backup and restore operation will alleviate most fragmentation problems on a disk. However, 24 hours per day, 365 days per year (24x365) data center environments typically cannot afford to close entire applications to perform full image backup and restore operations.

A Digital layered software product, DEC File Optimizer for OpenVMS, lets you reduce file fragmentation on a disk while that disk remains on line. DEC File Optimizer for OpenVMS uses an OpenVMS subfunction called **movefile**. The product performs defragmentation activity across the disk on a file-by-file basis. Data integrity rules ensure that the **movefile** subfunction repairs a fragmented file only when that file is not being accessed. Repairing a fragmented file consists of moving the file fragments to a new location on the same disk such that the file resides contiguously.

DEC File Optimizer for OpenVMS:

- Reduces file fragmentation.
- Enables you to select or to exclude files based on:
  - User choice
  - How badly the files are fragmented
  - Whether the file has placement pointers
  - Whether the files are indexed
- Provides for scheduling of defragmentation jobs.
- Enables you to classify files as either frequently or infrequently accessed and to have them placed accordingly. For instance, frequently accessed files can be placed near the file system's index file so that the physical movement necessary for the disk's read/write head to travel from the index file to the actual file is decreased. In addition, depending on the application, certain files that are frequently accessed can be clustered near one another if the same application uses them in an intermixed fashion. Infrequently accessed files can be placed near the high **logical block number** (LBN), near the end of the disk, or clustered near other files that are infrequently accessed.
- Operates in AXP and VAX environments.

OpenVMS provides the ACP-QIO **movefile** subfunction that moves the contents of a file, or part of the contents of a file, to a new disk location. You can use this subfunction to build your own disk defragmentation application.

A program can invoke a **movefile** subfunction by issuing a QIO request using the function code IO$_MODIFY and the function modifier IO$M_MOVEFILE. The **movefile** subfunction moves a file, or specified consecutive virtual blocks within a file, to a new location. This subfunction applies only to disk volumes.

A program can perform a **movefile** operation on a file if the following conditions are met:

- The program has write access to the file.
- The file is closed.

- **Movefile** operations are not disabled on the file.

  Movefile operations are automatically disabled on critical system files. You can disable **movefile** operations on specific user files by specifying the /NOMOVE qualifier with the SET FILE command.

- The **movefile** operation is not interrupted.

  If the **movefile** operation is interrupted by any other operation, such as a read or write operation, the **movefile** operation aborts and the file remains in its original position. This safeguard protects the integrity of your on-disk data and lets you safely attempt regular disk defragmentation operations without the burden of shutting down all access to a fragmented disk.

- The source and target locations are on the same disk.

  You cannot transfer blocks from one disk to another and you cannot move blocks spanning more than one disk.

See the *DEC File Optimizer for OpenVMS Guide to Operations* for details about DEC File Optimizer for OpenVMS. Refer to the *OpenVMS I/O User's Reference Manual* for information about the **movefile** programming interface.

9.7 OpenVMS Batch and Print Recovery Techniques

The OpenVMS batch and print queuing system allows the queue manager and the job controller to function separately. This design provides the following distinct dependability advantages:

- Automatic queue manager failover in VMSclusters

  A single queue manager process acts as a clusterwide server, accessing the queue database for all processes in a VMScluster. Job controllers, user processes, and symbionts all communicate directly with the centralized queue manager through a shared IPC link.

  The START/QUEUE/MANAGER command is a clusterwide command that starts up a single queue manager process that provides queuing services for all nodes in a VMScluster. By default, any node in the cluster is eligible to run the queue manager process. That is, if a node on which the queue manager is running leaves the cluster, the queue manager can fail over to the next available node. You can specify a node list in the START/QUEUE /MANAGER command and, if desired, limit the nodes to which the queue manager can fail over.

  During a queue manager failover, queues are not stopped. All requests to the queuing system complete as expected.

- Execution queue failover with autostart queues

  To increase the availability of queues in a VMScluster, you can specify a list of nodes (for batch queues) or nodes and devices (for print queues) on which an execution queue can run. You can use the ENABLE AUTOSTART/QUEUES command to provide the ability for the queue to fail over to another node if the node on which it is running leaves the cluster. You should use the autostart command on all nodes with queues that must be highly available.

  Using the autostart features does not guarantee that jobs will survive system crashes. For example, if an autostart queue fails over, the job that was executing in the queue is requeued only if it is restartable. Print jobs are restartable by default, but batch jobs are restartable only if they are submitted with the /RESTART qualifier.

When a node reboots, autostart is disabled until you enter the ENABLE AUTOSTART command. You can also add the ENABLE AUTOSTART /QUEUES command to your system startup procedure directly after the startup commands that configure printer devices and mount important disks.

- Restarting batch jobs

  If the system fails while your batch job is executing, your job does not complete. When the system recovers and the queue is restarted, your job is aborted and the next job in the queue is executed. However, by specifying the /RESTART qualifier when you submit a job, you indicate that the system should reexecute the job if the system crashes before the job completes.

  By default, a batch job is restarted beginning with the first line. It is possible to specify alternate starting points within a procedure so that you do not reexecute parts of the job that have completed successfully. Instructions for specifying different starting points in a batch job are discussed in the *OpenVMS System Manager's Manual: Essentials* and the *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems*.

  In addition to restarting a job after a system crash, you can also restart a job after you explicitly stop the job. To stop a job and then restart it on the same or on a different queue, use the STOP/QUEUE/REQUEUE/ENTRY command. For example, a batch job number 212 is stopped and restarted using the STOP/QUEUE/REQUEUE/ENTRY=212 SYS$BATCH command.

Refer to the *OpenVMS System Manager's Manual: Essentials* and *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems* for details about the system management tasks and commands associated with the centralized queue manager and its failover capabilities. See *OpenVMS System Services Reference Manual* for programming information related to this feature.

9.8 Implementing the Security Policy of the Data Center

The terms dependability, reliability, and fault tolerance take on new meanings when the faults mights be caused maliciously. Security policies can have an impact on the dependability and the viability of a business depending on how easy it is to get around the security that is built into the operating system, the data files and the database systems. Any security breach, even one as simple as a user unintentionally deleting a key file, can result in system or application unavailability.

There is a broad spectrum of security breaches. The breaches range from not allowing anyone to accidentally or maliciously take down your system, to providing a proper password policy that controls access to the system, to limiting the allocation of resources so that users do not unintentionally "give" the system to unauthorized users. It is best to establish a security policy that outlines potential security problems and security rules. Then follow the guidelines you defined when you assign new user accounts, passwords, default file protection, and user access.

Generally, implementing a security policy is the responsibility of the system manager because providing a secure system requires a knowledge of the business applications and a thorough understanding of how to control accounts, passwords, privileges, and quotas to keep the data center secure. Once security policies are established, operators and terminal users must implement proper procedures for handling data to ensure integrity and security. Even when security is not a major concern, you should establish some security policies to prevent innocent operator or user mistakes from unintentionally impacting the security of the data center.

OpenVMS provides a **C2** level of security. OpenVMS VAX Version 6.0 and later releases are certified by the U.S. government as C2 compliant. OpenVMS AXP Version 6.1 contains the same C2 security features as the formally certified OpenVMS VAX, although OpenVMS AXP Version 6.1 is not certified at this time.

OpenVMS security resists invalid access through:

- A user entry and password system that includes these login enhancements:
  - Forced hangups on multiple login failures
  - Break-in detection and disabling of accounts for a period of time after detection of a break-in attempt.
  - Automatic account expiration
  - Account restrictions based on time of day and day of week.
  - Restrictions on type of login, for example, allowing only local logins and disabling dial-up or network access.
  - Password vulnerability that can be minimized by establishing a minimum password length and specifying a password expiration period. You can also use a random password generator, which creates lists of nonsense words from which users can choose a password.
- An operations log that includes provisions for alerting security staff of ongoing security breaches.
- An encryption system that provides a high level of security for sensitive files. VAX Encryption is an OpenVMS layered product that performs encryption and decryption of disk-resident files.
- Discretionary access control lists (ACLs) and user identification codes (UICs) that permit individually named users to be either included or excluded from accessing a file or achieving certain forms of access (that is: READ, WRITE, EXECUTE, DELETE, CONTROL).
- A reference monitor that uses access control lists (ACLs) to protect files and other objects.
- Overwriting disk space before reassigning it to another user in order to prevent one user from obtaining disk space that contains another user's data.
- Notifying data center personnel or security officers of security events, and maintaining an audit trail of such events.

The security features can work, however, only if data center personnel use them and follow security procedure guidelines to enforce the proper constraints on user accounts and system account passwords. User account privileges, quotas, and access rights must be established to prevent users from taking actions that could impact the system integrity.

After you set up a security policy, you should enforce it using some form of auditing. You can audit by writing procedures and submitting them as batch jobs that regulary check for unusual system activity. You can also set up and use auditing tools to log events that are security related. Auditing is an important aspect of security. By regularly analyzing activity on the system, you ensure that security measures are not broken.

Network security is more challenging than security on a standalone system. One reason for this is because OpenVMS runs on all AXP or VAX systems from one generation of technology to the next. That is, network security is more complex because the compatibility between AXP or VAX systems increases the operational complexity at the same time it decentralizes system control thereby subjecting data to interception across the network.

Network security is of particular concern in the area of creation and management of the overall authorization database. An advantage of the AXP, VAX, and OpenVMS system compatibility is that you can implement security features across a network of AXP and VAX systems.

Through the use of Ethernet security features, proxy accounts, data encryption, the security modems such as DECmodem and PATHWORKS products for security of personal computer (PC) files, the OpenVMS operating system can afford a high degree of security on distributed systems.

The *OpenVMS Guide to System Security* contains detailed information about evaluating your security requirements and custom tailoring a security policy for your system and network. The guide also discusses using the DCL command SET AUDIT, which allows you to manage the audit server and the Audit Analysis utility (ANALYZE/AUDIT command) to help you analyze audit log files.

For help from security consultants, Digital offers the OpenVMS Security Review Service (VMS SRS) and the OpenVMS Security Enhancement Service (VMS SES). These services may help you evaluate and improve your current level of system security (see Chapter 10).

9.9 Supporting a Distributed Environment

Creating the necessary support system for controlling the data center is a major concern in an environment with distributed workstations and personal computers. When you put a workstation on a network or in a VMScluster, the potential exists for the naive users to impact the reliability of the entire data center.

The system manager can use SYSMAN (the System Management utility) or the DECnet System Services (DSS) to execute commands on a stand-alone node or across nodes and clusters. Using a utility such as SYSMAN helps centralize system management so that remote system management chores (such as the installation of software, backup, or account management on workstations) are not left to workstation users.

Consolidating system management so that the system manager can have a clusterwide view of the system from his or her terminal leaves less chance that the users will introduce unnecessary system problems. For example, a workstation user might upgrade to the latest version of some application and cause incompatibilities in a common database. Because the workstation user's actions might not be obvious, attempts at solving the problem could lead to extensive down time and unnecessary repairs.

Distributed system support services are also available through DECnet System Services. These services include the VAX Distributed File Service (DFS), the VAX Distributed Name Service (DNS), the VAX Distributed Queue Service (VAXdqs), and the Remote System Manager (VAXrsm).

9.10 Supporting VMScluster System Environments

Using the OpenVMS operating system in a VMScluster system environment is, for the most part, the same as using the OpenVMS operating system in a single node or networked environment. The following sections address some additional areas that you should consider for optimal system operation. Refer to Section 7.6 for related information about the VMScluster DECnet alias.

9.10.1 VMScluster Quorum Disk

One of the mechanisms used by VMScluster software to ensure that access to data on common devices is correctly coordinated is called **quorum**. By requiring that more than one-half of the (nonsatellite) nodes in a VMScluster configuration have established proper SCA communications before any node is allowed to continue, a pathological condition called a **partitioned cluster** is avoided. In a partitioned cluster, one group of nodes "thinks" it is the cluster while another group thinks the same thing but neither group is aware of the other group. However, each group is writing on commonly accessible disks. Refer to *VMScluster Systems for OpenVMS* for details on how to set up your VMScluster quorum-related parameters properly to prevent partitioning.

In the case of a two-node VMScluster configuration, the quorum calculation becomes problematic. If each AXP or VAX node had one vote (as recommended), and one node failed, quorum would be lost and the VMScluster would hang (to protect data integrity) until quorum is restored (that is, a vote count greater than one-half the total). To circumvent this situation, the VMScluster software can be configured to recognize a file on a commonly accessible disk as having a vote of one. Therefore, in the case of a two-node VMScluster, the total vote becomes three, with quorum equal to two votes. When one node fails, two votes still remain (one for the CPU node and one for the quorum disk) and the surviving node can continue to provide service.

The VMScluster quorum disk cannot be a member of a shadow set. OpenVMS Volume Shadowing cannot operate unless the OpenVMS distributed lock manager is available. The OpenVMS distributed lock manager cannot execute if quorum has not been established, and therefore it is impossible to make the VMScluster quorum disk a member of a shadow set. An OpenVMS system disk can be shadowed but the quorum disk cannot.

The VMScluster software uses only a particular file on a quorum disk. Thus, that disk may also be used for other purposes.

Because VMScluster configurations with three or more nodes do not require a VMScluster quorum disk, one valid option in the context of a dual-host configuration is to set up a VMScluster quorum node. By giving a third AXP or VAX node a vote, the quorum calculation works out without the need for a quorum disk. You must boot the third node from its own system disk, not from the boot members' disks. In this way, the quorum node can boot independently to maintain or to restore quorum as necessary.

9.10.2 VMScluster Common System Disks

With the appropriate definition of logical names, the VMScluster can use a single copy of most system files and images. This rule is per architecture; that is, all the AXP nodes use operating system files on an AXP system disk, while all the VAX nodes use operating system files on a separate VAX system disk.

The various VMScluster member nodes can each load the COPY.EXE image from the same directory on the same commonly accessible disk. This can save considerable disk space (especially for VMSclusters with many nodes) and is easier to manage consistently than completely separate system disks for each node. Refer to *VMScluster Systems for OpenVMS* for information on how to set up your VMScluster system to use cluster-common system disks (one per architecture).

The disadvantage of using cluster-common system disks is that the I/O traffic on the VMScluster system's common system disk may become too great to handle with a single drive. It also becomes even more important to shadow the system disk to avoid a single critical point of failure in the VMScluster system.

9.10.3 Multiple OpenVMS Versions (Rolling Upgrade)

The OpenVMS operating system allows a VMScluster system to continue to provide service while the system software is being upgraded to the next release. This process is called a **rolling upgrade** because each node is upgraded and rebooted in turn, until all the nodes have been upgraded.

During the period when two versions of OpenVMS are running on the VMScluster, there are at least two system disks (or at least two system directory trees on a disk). Each has its own version of OpenVMS and each node has booted OpenVMS from one of those system directories. The existence of system disks with different versions of OpenVMS requires special consideration for VMScluster configurations with satellite nodes.

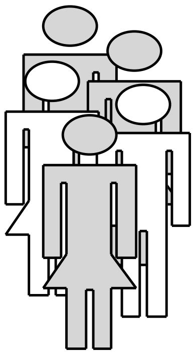
When a satellite loads its copy of OpenVMS from a boot member, a special network program called a **load assist agent** is used to facilitate the process. Because this program controls the booting protocol, its version should match the version of OpenVMS being loaded into the satellite.

In a VMScluster where the boot members are running different versions of OpenVMS (for example, when the VMScluster is in the middle of rolling forward), you may have some satellites that are running the old version of OpenVMS and some that are running the new version. Once all the boot members have been upgraded to the new version, however, you may not have a problem. If the booting protocol has not been changed between OpenVMS versions, the satellites that want to load the old version may continue to do so, as long as you keep their old system disks accessible to the boot members.

If the booting protocol has in fact changed between the old and the new versions of OpenVMS, the load assist agent (running as part of the boot member's OpenVMS system) may not properly handle the loading of the old version of OpenVMS to a requesting satellite. Remember that all the boot members have been upgraded to the new version of OpenVMS at this point. This situation requires that you change the boot sources of the satellites to load the new version of OpenVMS.

Because it is not possible to predict the technical reasons for changing the boot protocol, the safest approach is to assume that it has changed. Therefore, the suggested practice is to plan for all satellites to load the new version of OpenVMS by the time the last boot member has been upgraded to that new version.

10 Dependable Consulting from Digital



In the final analysis, it is people who build dependable systems. You and your staff do not have to do it alone. This chapter describes a variety of Digital consulting services. You may elect to use these services to help you build dependable OpenVMS based computing systems when you do not have the expertise, time, or resources to do it alone. See the bibliography in Appendix B for information about a Digital consulting reference guide that your Digital representative can order for you. The guide describes in detail the services outlined in this chapter.

10.1 Application Characterization and Predictive Sizing Consulting

The application characterization and predictive sizing (ACPS) consulting services can assist you in analyzing computer resource requirements of your business applications. These services provide a methodology that allows you to relate specific business units of work with hardware requirements for the purpose of sizing and capacity planning. Your applications need not be currently executing on a Digital system for characterization. Prior to any coding of new applications or migration or porting of current applications, the ACPS services can provide capacity predictions that will improve your chances of successful implementation of your new system.

The following consulting services are available:

- Business applications needs analysis
- Predictive sizing
- Application characterization
- Application level capacity planning

Various Digital service tools, such as First Approximation Sizing Technique (FAST), Application Characterization and Evaluation (ACE), FAST-Q, and Estimator for System Performance (ESP) are used in the delivery of these services to provide comprehensive information on which the Digital consultant may base conclusions.

10.2 Capacity Planning Consulting Service

Capacity planning is the prediction of system resources required to support the future computing needs of the business. It provides means to decision makers for the timely acquisition and provision of system resources capacity.

The capacity planning consulting service covers the following:

- Assessment of current performance and service levels.
- Analysis of workload and service level requirements.
- Saturation analysis and workload balancing.
- Modeling of alternative AXP and/or VAX configuration options to support scenarios as defined by the customer.
- Evaluation of AXP and/or VAX configuration(s) and future growth path(s).
- Presentation of the results. The executive summary of the report (written for a nontechnical audience) is the basis for a presentation and discussion with the customer's management.

10.3 COHESION Support Consulting

COHESION is the name for the Digital complete computer-aided software engineering (CASE) environment. There are several services available to help you take advantage of the COHESION environment.

CASE program services that support COHESION:

- Visionary planning

  Digital management consultants establish the management framework for guiding the entire COHESION implementation process.
- Critical success factors

  A management workshop to help ascertain those factors that are critical to the successful achievement of your strategic objectives.
- Software assessment

  A formal look at your development organization's strengths and weaknesses using a methodology called SEI Software Process Assessment Consulting Service. Digital has been certified by Carnegie-Mellon's Software Engineering Institute to perform this service.
- Implementation planning

  Identifying the proper set of activities and tasks necessary for a successful project implementation.
- CASE environment design and planning

  Digital consultants help design the S/W Development Life Cycle (SDLC), which becomes the framework for designing an integrated COHESION environment.
- CASE technology selection

  Establishes business justified selection criteria for determining which CASE tools make sense.
- Concept education

Classroom, seminar, and video format training in Basic Software Engineering Principles, as a prerequisite to implementing a CASE program.

- CASE tools training

  A wide variety of training classes on CASE tools. If desired, a customized training program can be designed and implemented.

- CASE tools consulting

  Digital consultants help your organization raise the experience curve in advanced usage of CASE tools.

- DECstart™ for DECset

  Installation and customization help for the CASE tools.

- Implementation reviews

  Digital consultants provide ongoing monitoring of the implementation and recommend ways to stay in tune with business goals.

10.4 Contingency Planning Assistance Consulting Service

This service identifies organizational risks associated with loss of information systems due to catastrophic incidents or other unplanned extended outages. Recommendations address disaster recovery strategies to protect your business investments.

10.5 Customer Training Advice Package

This section covers the customer training advice package.

10.5.1 Course Formats

Digital provides a variety of course formats that allow customers to choose where they learn, when they learn, and how they learn. Customers can choose self-paced courses, hands on classroom training with practice labs, lecture with case study, and so on. Formats for customer training include:

- Lecture/lab
- Seminars
- On-site training
- Digital Press documents
- Cooperative training program

10.5.2 Purchase Options

Digital offers several packaged training programs to discount and simplify the training purchase. All of the packages include telephone or in-person consultation for determining what courses should be taken.

Packaged training programs include:

- Unlimited training subscription
- Eduplan training subscription
- DECplan™ training account
- REDuplan training subscription

- Complementary solutions through training

10.5.3 Comprehensive Training Solutions

The Comprehensive Training Solutions (CTS) group creates customized solutions to meet the needs of an organization's unique work environment, the current capabilities of its employees, and the skill and knowledge requirements of their new jobs. For example, customers choose CTS when they need custom training on an application, job-specific training delivered to the job site, computer room-to-desktop training coordination, training on technologies and applications from vendors other than Digital, management and business skills training, and turn-key training management and administration.

CTS services include:

- Enterprise human resource consulting
- Strategic assessments
- Training needs analysis
- Project management
- Courseware maintenance
- Computer-based instruction
- Print and online job aids
- Manuals and documentation
- Multimedia courses
- Introductory videotapes
- Computer-based interactive video
- Classroom instruction and workbooks
- Customer online documentation screens

CTS enables customers to adapt to changes in their business. CTS can design comprehensive training solutions that make work simple for employees and that make enterprises successful.

10.6 DECstart Consulting

DECstart services deliver a defined set of tasks at the customer's site, structured to ensure an efficient startup of the system. DECstart familiarizes you with the software and provides orientation and hands-on experience for your staff. There are DECstart services for several major software products, including:

- ALL–IN–1 DESKtop for DOS
- ALL–IN–1
- ALL–IN–1 Version 2.3 Upgrade
- DECnet for OpenVMS
- Local area VAXcluster
- Digital NMCC/DECnet Monitor
- PATHWORKS for Macintosh
- PC packaged installation service

- PCSA (PATHWORKS for MS–DOS)
- ULTRIX (VAX and RISC)
- VAXcluster (VAX only)
- VMScluster (AXP only, or AXP and VAX)
- DECset (DEC LSE/SCA, VAX DEC/CMS, VAX DEC/MMS, DEC Test Manager, DEC PCA)
- VAX SPM™
- OpenVMS Version 6.*n* upgrade
- OpenVMS

10.7 Digital Program Methodology Consulting

These programming methodology services involve a phased approach to understanding requirements and to carrying out the appropriate development activities to satisfy your programming needs.

There are several ways that Digital programming methodology services can work collaboratively with your staff:

- Conversion services
- Custom application modification
- Custom project service
- Custom service package
- Custom software application
- Design phase service
- Functional specification service
- Requirement specification service
- NETplan™ application design service
- NETstart™ application development service
- NETplan design analysis service
- NETplan protocol design service
- NETstart protocol development service
- NETplan requirements analysis

10.8 DSNlink—Customer Access to Existing Knowledge Databases

DSNlink is an OpenVMS based software tool that allows on-demand computer-to-computer communication between customers and Digital Customer Support Centers. DSNlink is being offered to OpenVMS customers who currently receive telephone support.

DSNlink is free of charge to contract service customers. In order to use the software, you are required to sign and return a software license. Receipt of the license agreement will trigger the distribution of DSNlink media and documentation and an authorization code.

You will need an additional telephone line and a 2400 baud modem in order to use DSNlink. Digital pays for the phone line connection to the Customer Support Center but not for the line itself.

Substantial improvements have been made to the symptom and solution database. More than 40,000 technical articles are contained in the database with more articles being added daily.

Bu using advanced techniques in data compression, DSNlink can provide outstanding throughput–twice the throughput of the rated modem speed. DSNlink ensures the security of both the system and the data passing over the service network by employing the latest security features such as system authentication and an allocated OpenVMS port.

The following applications are now available to contract customers who install the software and have a phone line and a modem:

- Consulting Service request submission
- Remote database search
- File transfer facility
- Flash message facility

10.9 Enterprise Integration Centers Advice Package

In addition to the standard Digital products that the Enterprise Integration Centers (EIC) provide through the various product families, EIC provides custom hardware and software engineering, production and integration capabilities. Custom hardware design ranges from modification to Digital or other products to complete custom product design. Custom systems include custom configurations that meet special customer requirements for space or environment.

EIC will also procure, integrate, and test third-party equipment as part of the overall system to meet the customer's application needs.

The technology areas in which EIC has expertise are as follows:

- Remote terminal connections
- High performance communication interfaces
- High availability systems
- Real time systems
- Factor floor hardware
- Impact line printers
- Tape and disk systems
- Image products
- Voice products and systems

10.10 Enterprise Planning and Design Consulting

These services are intended to assist you in formulating high-level business system strategies, design, and architectures. They can be formulated to have a business, information technology, and human factors focus.

Enterprise planning and design planning services include:

- Strategic planning
- Strategic design
- Information architecture
- Solution architecture

10.11 Help Desk Consulting Service

The help desk service offers you a range of service for the design, implementation, and management of computer help desk operations. The service is designed to cover multivendor environments and address your entire computing environment, regardless of which vendors' hardware or software you use.

There are three primary modules in this service:

- Evaluation and design
- Implementation and operation
- Problem resolution coordination and management reporting

10.12 Migration Consulting

These services are designed to minimize the cost, risks, and work needed to successfully complete any migration project and to ensure an unlimited growth path for the future. These services include:

- Migration planning, design, estimation, implementation, and project management
- Specification and application design and documentation development
- Customization of software conversion tools
- Support for third-party products and conversion tools
- Help for establishing parallel system runs
- Development and assistance with acceptance testing and implementation
- Integration of existing equipment
- System performance analysis, tuning, and capacity planning

10.13 Network Performance Consulting

Network performance consulting provides you with an evaluation of your network design. The consultants provide tools, methodologies, and expertise in network performance to address your long-term network usage needs. Networks ranging from medium-sized local area networks to geographically disperse Ethernet networks can be analyzed.

There are three types of network performance consulting:

- Network performance analysis
- Network performance optimization
- Network performance management

10.14 Packaged Application Software Solution Consulting Service

Packaged application software solution service includes the applicable packaged application software and consulting services necessary for the installation and operation of the packaged application software. Packaged application software solution service is provided in accordance with the provisions of Digital standard terms and conditions.

10.15 Professional Consulting Services

Professional services are intended to provide you with professional resources to carry you through periods when you need or cannot find the appropriate talent to hire.

There are several types of professionals whose services you may acquire:

- Acquiring services and support
- Administrative support
- Consultant I, II
- Programmer
- Program manager
- Network consultant
- Network systems engineer
- Senior consultant
- Senior systems engineer
- Software engineer
- Senior consultant I, II, III, IV
- Systems engineer

10.16 Recover-All Consulting Service

The recover-all service supplements the Digital on-site maintenance agreements and provides priority repair or replacement of your Digital and multivendor equipment damaged in an accident or disaster. It includes reimbursement for many expenses associated with getting your computer operation functioning at full capacity, including a temporary data processing location, removal of equipment, and replacement of certain software. No deductible or depreciation is applied to your equipment.

Recover-all service covers damage caused by:

- Power failure
- Air-conditioning malfunction
- Fire

- Earthquakes, flood, and other natural disasters
- Theft
- Vandalism
- Accidental damage from beverage spills and other accidents
- Sprinkler leakage, sewer backup, and plumbing accidents

10.17 RESTART Consulting Service

The RESTART service provides two alternate computing centers that let your business operations continue to function during a disaster or other extended outage. Eleven business recovery centers in the U.S. extend access to hot standby sites by providing secured office facilities for recovery staff during a disaster.

10.18 Systems Integration Advice Package

Under the systems integration advice package, Digital will serve as a single point of contact for planning, designing, implementing, or managing your computing system. The service provides for integrating hardware, software, and services to address your business needs and organizational requirements. The computing systems can be from Digital, other vendors, or both.

Relying on a systems integrator to manage information technology efforts lets you focus your resources (staff, skills, dollars) on your primary business.

10.19 VAX Performance and Capacity Consulting Services

The VAX performance and capacity services (VPCS) provide a methodology for planning strategic system growth. VPCS brings Digital tools, expertise, and methodologies to help you plan system capacity growth.

There are three different services in VPCS:

- Performance analysis service
- Performance optimization service
- Performance management service

10.20 OpenVMS Security Enhancement Consulting Service

The OpenVMS security enhancement service (OpenVMS SES) provides mandatory (nondiscretionary) access controls and security auditing features for OpenVMS. It is designed for the customer who must maintain a secure data processing environment involving multiple levels and categories of classified data and users with varying security clearances. This service provides the system manager or system security officer with a toolset to devise a system-wide security policy that safeguards users, data, software, and hardware from security threats.

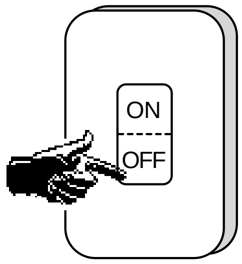
OpenVMS SES augments the numerous security features already present in the OpenVMS operating system. It provides additional features to meet the rigorous security needs of government agencies, national defense organizations, prime contractors and other customers that must label and protect classified information processed on OpenVMS systems. It also provides a means to evaluate the effects of mandatory access controls on application design and system management.

OpenVMS SES consists of two components: a set of tasks to be performed by Digital at the customer's facility and packaged software to be installed by Digital on the customer's computer.

10.21 OpenVMS Security Review Consulting

The OpenVMS Security Review Consulting Service is an evaluation of the customer's established security controls. The key component of the OpenVMS Security Review Consulting Service is the consulting service provided by Digital. This service provides the system manager or system security manager with a list of findings along with recommendations, if appropriate, that can be implemented to improve the system security.

11 Case Study: Lights Out Data Center



This case study describes the implementation of a highly dependable data center at the Digital Customer Support Center (CSC) in Colorado Springs. The CSC's business is to solve technical software and hardware problem reports submitted by customers. The service is provided for those customers who have a support contract with Digital and is available 24 hours a day, 365 days a year (24x365). Starting in 1984, the CSC data center personnel launched an aggressive plan to meet CSC's business requirement for uninterrupted computing services. The result, which continues today, is a highly automated style of computing operations called a **lights out data center.**[1]

In a lights out environment, computing functions and components are so well automated and monitored that minimal operator or system manager intervention is needed. In theory, no human operators are needed in the physical data center to maintain operations. Thus, the concept developed of being able to turn off the data center's lights while the computing equipment operated unattended. In practice, most lights out data centers require occasional maintenance by human operators. However, a carefully designed and tested lights out data center truly does require far fewer operators and typically experiences far less system down time. Of course, the issue of whether the lights are literally *on* or *off* is not important. The significant feature of lights out computing is that it allows for a separation of data center staff from the computing system hardware.

The lights out computing style can cut down on the cost of operating large, critical labs. When an early stage of a computing problem is detected, software alarms initiate corrective action. If the problem cannot be fixed automatically, the monitoring software notifies the appropriate person via predefined mechanisms. Examples of automated notifications include activating pagers (or beepers) worn by selected personnel, starting recorded DECtalk announcements on the facility's loud speaker system, and sending electronic mail messages to the VAXcluster manager.

The following sections explain how the CSC data center personnel:

- Changed their own perception about the work they performed and about the people and jobs their computing services supported.
- Worked closely with members of their staff who were affected by a decision to eliminate costly and menial computing operation tasks.
- Eliminated most tasks that depended on manual intervention by a large staff of operators who had worked three shifts.

[1] Digital operates other lights out data centers. This case study focuses on a specific implementation at the CSC data center in Colorado.

- Developed an effective partnership with the staff of the building facilities organization.
- Developed an unofficial but still effective partnership with city and utility planners to ensure that redundant power, telephone, and water systems were available.
- Evaluated the products and tools available to the CSC data center at the time (starting in 1984) and selected technologies that were flexible and could grow with industry improvements to software and hardware components.
- Built their own sophisticated alarm system that detected when computing problems were about to occur or were occurring. The software system automatically activated defined sets of preventative or remedial steps. The CSC data center's alarm system evolved to become a product called DECalert.

Note

In February 1994, Digital granted Target Systems Corporation a nonexclusive, nontransferable license to use DECalert Version 1.2 software. Under the agreement, Target Systems may provide support for the Digital customer base (existing and future) for the product, as well as future development and migration of the product or its derivatives. Target Systems named the product TARGET–>ALERT.™ Digital reserves the right to continue to sell DECalert or its enhancements, as Digital deems fit.

- Ultimately, achieved 100% application availability and the confidence of the data center's users.

As members of the CSC data center's staff would say:

- "We are like all other data centers."
- "We are like some other data centers."
- "We are like no other data center."

Keep these quotations in mind as you read this case study. The steps taken by the Digital CSC data center group in Colorado Springs to ensure 100% application availability are appropriate for their buildings, geographical surroundings, applications, communications needs, and personnel. You probably will recognize many or some of the problems that were resolved by the CSC data center staff. At the same time, your business is unique and the CSC's solutions may be inappropriate at your facilities.

Use the CSC data center's experiences as background information. This knowledge may be helpful as you improve the data center at your site to meet your business requirements.

11.1 Customer Support Center Business

The Customer Support Center (CSC) data center provides computing services to the CSC business, which solves technical problem reports submitted by customers of Digital. The CSC business used the computing resources for a variety of functions. A few examples include:

- Recording, tracking, and resolving software performance reports submitted by customers

- Communicating with Digital engineering organizations via a worldwide, company-wide data network
- Reproducing the combination of software product(s) and hardware equipment that resulted in an error, as reported by customers of Digital

11.2 Time for a Radical Change

In 1984, the computing services staff for the Customer Support Center (CSC) in Colorado Springs knew that they had to solve a very difficult problem. The CSC business needed to operate in 24x365 mode to meet Digital customer needs for continuous telephone technical support. Between 1979 and 1984, the computing services that had been provided for the rapidly growing CSC were best described as unpredictable and chaotic. A few examples include:

- Systems were shut down with little advance notice to the user community.
- Software products were not tested adequately prior to installations or upgrades to determine their potential impact on existing applications.
- Carefully-planned fault prevention strategies did not exist.
- Error correction strategies were haphazard.
- Failure recovery plans were not well organized.
- Protections against and contingency plans for the impact of environmental factors (like power outages in Colorado Springs) at the facilities either did not exist or were not adequately developed.

The pre-1984 environment of the computing services data center was later characterized as "...a traditional MIS shop". The staff of 62 persons seemed to be very busy but their activities were almost constantly *in reaction* to problems as they occurred (react mode). Many staff members at the pre-1984 data center worked hard and had the best of intentions to deliver quality computing services but there never seemed to be enough time to evolve into a proactive, well-designed, dependable data center.

By 1984, CSC business demands and exceptional leadership forced changes at the CSC data center. Several leaders stepped forward and declared, in effect, "We need a new vision." The leaders assessed their current (1984) organization and noted the following:

- They had 62 persons working at the data center.
- The cost of operating the data center (equipment, personnel, rent, utilities, plus other factors) was a significant part of the overall CSC budget.
- Staff members generally worked in just one of four highly specialized fields:
  - Operations
  - Application development
  - System management
  - Application support

  Members of each functional organization knew practically nothing about the jobs, strategies, and technologies of fellow employees within the three other CSC data center functions.
- There were multiple data centers supporting Digital facilities across Colorado Springs.

- There were multiple standalone hardware configurations.
- There were multiple operating systems.
- Supporting the data centers was *people intensive*. That is, when problems occurred, staff members were detached to isolate and fix it. Preventative, diagnostic, and remedial tools (software and hardware) did not exist.
- The data center had difficulty responding to changing business requirements of the CSC business.

To further understand how the CSC data center operations might be improved, one of the leaders of the data center's availability project changed hours. This system manager switched from the first shift[1] to the third shift[2]. Most of the existing operator's tasks, like nightly backup operations, occurred on the third shift. The system manager spent three months on the third shift and developed ideas about lowering staffing costs and changing the mindset, in a positive way, of the people who performed the operator functions.

11.3 Customer Expectations

The Digital Customer Support Center (CSC) data center staff leaders conducted surveys, interviews, and meetings with the people who used the computing services to learn about their expectations. The effective communication between the data center and the CSC business was an immediate and highly visible change in the way the CSC data center performed its business. Also, the data center decided to deliberately refer to their users as "customers." This change was meant to convey an understanding by the CSC data center staff that their jobs depended on the satisfaction of the people who used their services.

CSC *customer* expectations recorded during this early (1984) phase included:

- Lower costs.
- 100% *application* availability. Note that CSC personnel did not focus simply on hardware availability. For instance, if a CPU board failed but the data center computing systems were configured in a way that allowed CSC's applications to continue, then the CSC business could continue operating without interruption.
- Support growing user and application base.
- Be highly responsive in terms of application response time, problem resolution, and enhancements.
- Be flexible to changing CSC business needs and individual needs.
- Maintain security.

To meet the expectations of the CSC people who used their computing systems, the CSC data center leaders worked together to formulate the strategies described in Section 11.4.

---
[1] Generally, 9 a.m. to 5 p.m.
[2] Generally, 1 a.m. to 9 a.m.

11.4 Strategies for Achieving 100% Application Availability

To achieve 100% application availability, the data center leaders decided to develop strategies for six key components of the data center:

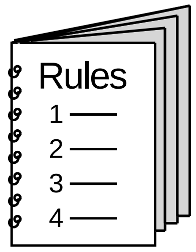
- Process
- People
- Hardware
- Software
- Environment
- Telecommunications

This section outlines the strategies the data center staff defined for these components. Section 11.5 describes how the staff implemented the strategies to achieve 100% application availability.

In this section and Section 11.5, the discussions are not exhaustive because:

- Refinements occur constantly, which allow the CSC Lights Out Data Center to maintain their advantage with current technology. As this handbook emphasizes, *dependability is a journey, not a destination.*
- Certain Digital proprietary technical details are intentionally omitted from this case study to protect the Digital investment in its 24x365 telephone support business.

11.4.1 Process Strategy to Meet CSC Business Needs

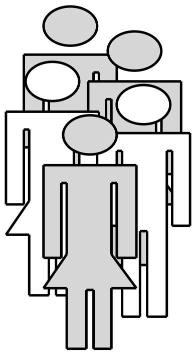


To support the CSC business requirements outlined in Section 11.3, the CSC data center staff decided that their process strategy must include the items in the following list:

- Automated backups (primarily disk-to-disk).
- A paperless environment in the main computing labs.
- Automated hardware monitoring and alerts.
- Automated software monitoring and alerts.
- Automated performance improvements.
- Automated system management.
- Automated utilities management.
- Automated layout (hardware configuration) management.
- Contingency plans for every imaginable event that could possibly interrupt service. For instance, Colorado Springs experiences snow blizzards that can last for days. Despite the severe weather conditions, customers of Digital who rely on the 800-number technical support still require 24x365 service.
- Documentation tools to assist in easier reporting of defined availability compliance rules.

- Project management software tools.
- Proactive response to computing system problems.
- As nearly as possible, a paperless environment in office areas.
- A new service request mechanism.
- A helpline or Help Desk with a call management system.
- Flexibility to meet changing business needs.

11.4.2 Staffing Strategy to Meet CSC Business Needs



The strategy for enhancing the data center's staff so that it could contribute to CSC's requirements for 100% application availability, as outlined in Section 11.3, were the items in the following list:

- Develop and encourage a cross-training policy

  Instead of each person knowing about only one specialized field (operations, application development, system management, or application support), implement cross training so that the CSC data center team would consist of senior-level generalists. Combine hardware, software, and applications expertise within the VAX and VMS environment. Each person remains an expert in one or several particular disciplines; however, their knowledge is broadened so that they are aware of the full impact their actions have on other processes and computing functions. Support this effort through funding and flexibility so that employees can receive the training and practical experience they need.

- Create support teams to improving responsiveness and flexibility

  The CSC data center staff should build support teams that address the immediate and longer-term needs of the following CSC areas:

  - Systems
  - Applications
  - Customers (CSC personnel)
  - Developers

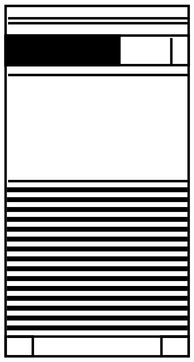
- Eliminate unnecessary tasks and jobs, limit menial tasks through automation, and help displaced employees find more rewarding jobs in the company

  The CSC data center's staff realized early on that there was "more to life than mounting magnetic tapes (for nightly disk-to-tape backups), fixing paper printers, and sorting line printer output." In the pre-1984 CSC data center environment, these tasks occupied a large percentage of the operators' time.

After working openly with its staff about its goal to lower operating costs, the CSC data center reduced its headcount from 62 to 38 persons in one month. Instead of dismissing the 24 displaced employees (11 of them were operators), the data center management counseled the displaced employees and helped them to consider more rewarding career opportunities, to pay for any training that they needed, and to start jobs in other Digital organizations.

The sensitive way in which the transition was handled by the CSC data center management led to positive feelings among the people who remained in the data center. Remaining people felt that their management not only had worthy and exciting goals of building a highly available computing service but it had treated their friends with dignity and care. The psychological benefits that derive from a fair, sensible policy like the one in this case study are significant.

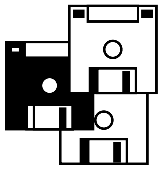
11.4.3 Hardware Strategy to Meet CSC Business Needs



To support the CSC business requirements outlined in Section 11.3, the CSC data center staff decided that their hardware strategy must include the items in the following list:

- VAXcluster technology to eliminate single points of failure at the node and disk level
- Remote management capabilities
- Predictive maintenance
- End node and routing networking capabilities
- Fault tolerance
- Capital-intensive characteristics rather than people-intensive characteristics
- Equipment that is secure
- Support from Digital Customer Services (for repair of hardware equipment)
- Flexibility in hardware configurations and capabilities to match changing business needs

11.4.4 Software Strategy to Meet CSC Business Needs



To support the CSC business requirements outlined in Section 11.3, the CSC data center staff decided that their software strategy must include the items in the following list:

- VMS operating system as the standard
- Automated fault notification software tools

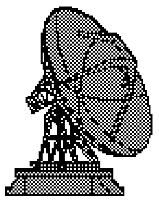
- Flexibility for changing business needs
- Software products that monitor security breaches
- Extensive online documentation of coding standards

11.4.5 Environment Strategy to Meet CSC Business Needs

To support the CSC business requirements outlined in Section 11.3, the CSC data center staff decided that their environment strategy must include the items in the following list:

- Uninterrupted service, protecting against events like power outages, fires, floods, snow and ice storms, and telephone vendor work interruptions (strikes)
- Cost effectiveness and flexibility for changing business needs
- Secure equipment and processes
- Remote access support

11.4.6 Telecommunications Strategy to Meet CSC Business Needs



To support the CSC business requirements outlined in Section 11.3, the CSC data center staff decided that their telecommunications strategy must include the items in the following list:

- Centralized monitoring and management of:
  - Hardware network components (stations)
  - Network operations like connections within local area network (LANs) and between the extended LAN and the wide area network (WAN)
  - Applications using the network
  - Security
  - Outage management
  - LAN and wide area network (WAN) management
  - Management escalation processes

    Having the authority to escalate data center issues to CSC management when an issue was not being addressed in the time frames set for a particular event.
- A video distribution system
- Ethernet and DECnet for LAN and WAN connections
- Remote access capability
- Performance, or throughput, adequate to satisfy the business functions of customers (CSC staff)
- Flexibility in the growth and dynamic changes to the network topology, to match changing CSC business needs

11.5 Implementing the Data Center’s Strategies

This section describes how the strategies discussed in Section 11.4 were implemented by the Customer Support Center (CSC) data center staff. Note that the strategy to fix problems in one area usually impacted one or several area(s). Achieving 100% application availability at the CSC data center did not happen by fixing the problems in category #1, people, and then moving onto category #2, hardware, and so on. Instead, the refocused CSC data center team worked to understand the dependencies within and between the six major components of dependable systems. They arrived at tentative solutions, carefully tested the effectiveness of each attempted solution, and remained flexible enough to consider alternative approaches at any point in their efforts to significantly improve the reliability, recoverability, and fault tolerance of their data center.

The Customer Support Center (CSC) lights out computing success is the result of integrating data center and application development processes with hardware and software configurations. Additionally, CSC made use of technologies that can monitor applications and data center operations. The strategies outlined in Section 11.4 translated into implementation tactics that can be categorized as follows:

- Human factors planning and management. See Section 11.5.1.
- Technology planning and utilization. See Section 11.5.2.
- Environmental support, which consisted of a new close partnership between the data center staff and the facilities staff to protect CSC against possibilities like power outages and other factors. See Section 11.5.3.
- Application development’s management and implementation. See Section 11.5.4.
- Overall operations support. See Section 11.5.5.

11.5.1 Sensitive Implementation of a Refocused Staff

Human factors planning and management played a key role in developing the CSC’s lights out environment. Staffing requirements were reduced and shifted away from computer operators and system managers to operational planning and generalist support personnel.

Responsibilities for the new generalist function included overall management of developers of applications, hardware low- to no-impact maintenance, support of the operating system, all applications, new and changing technology, the process, customer satisfaction, and capacity planning. In addition, there was an increased reliance on service vendors and organizations typically outside of data processing.

In the CSC’s *online* environment, the pressures of running a daily business plus migrating to a different computing strategy required maximum productivity, efficiency, and cooperation from all staff members. By developing and implementing a high-quality human factors management plan, the risk of staff interruptions was greatly reduced.

Early in the planning process, the data center found it very important to communicate the computing strategy *vision* to data center employees, CSC customers, and CSC executive management. In addition, executive management made clear to individuals their support concerning the shifts in job responsibilities and staffing levels so that they could remain confident in the company’s commitment to them. This was accomplished through career counseling and skills retraining.

During the planning or design process, plans for the continued computing migration strategy were communicated to the organization's employees. Feedback processes were established to solicit creative ideas and to identify potential problem areas. This expanded the information base that could be used as input to further refine business requirements. Project plans were one of the many business control processes used to implement and communicate change in the environment whether it was an application, system, telecommunication, service, or contingency.

11.5.2 Technology Planning and Utilization

The CSC data center staff found that technology planning and utilization was critical to a successful lights out implementation. Hardware technology had to support automatic, unattended system and network failover as well as remote fault monitoring and isolation techniques.

Digital VAXcluster configurations, which operated in a networked environment and were supported by appropriately planned OpenVMS operating systems with DECnet for OpenVMS, were the CSC's primary platform for lights out operations.

Additionally, several Digital technologies were used, including (what evolved to become) DECalert, VMScluster Console System (VCS), Remote System Manager (RSM), VAX Storage Library System (SLS), and DSNlink. These technologies perform many functions, including the monitoring of system, network, security, and application events. The events can trigger predetermined operational decisions or notify responsible personnel. See Section 11.6 for details about the CSC data center personnel's use of these products.

Hardware options such as printers, removable disks, and tapes were excluded in the CSC's lights out architecture. In place of these, distributed, customer-requested printing over the network, customer-requested report distribution, fixed media, high density disk units with required volume shadowing and off-site *write once, read many* (WORM) optical archiving were used. (This is an on-going implementation for some of the tape areas.)

11.5.3 Protecting Against Environmental Factors

An integral part of the CSC's planning process focused on the physical design and layouts of the facility, environment, and utility support. Electrical and HVAC support requirements were designed to deliver uninterrupted and properly conditioned service.

The process, which continues today, included a five-quarter forecast for facility modifications, systems, and telecommunications based on the business needs of the Customer Support Center. All electrical conditioning and HVAC equipment were designed to have automatic failover and remote monitoring/fault isolation capabilities.

The CSC's lights out data center was physically located away from high traffic areas and areas of the building that frequently undergo modification. Its location in the building optimized the availability of telecommunication and environmental utility support.

The layout on the data center also supported the expansion of existing clustered systems and minimized impacts to other production systems, maintenance of equipment, and cabling required for inter-system and network connections.

The design of the data center met all local and Digital construction and fire codes as well as Digital security standards. In addition, the design used remote environmental monitoring equipment.

11.5.4 Application Development Management and Implementation

In the CSC's lights out environment, applications running in the data center were not bound to any one system or site. Applications run across a VAXcluster and take advantage of dynamic load balancing and automatic failover capabilities, or run across sites. To develop these applications, phases from inception to production were used. Each one of these phases was supported by an internal operational support group.

During development, application developers were encouraged to establish standards that optimized operational management routines inherent within VMS, VAXclusters, clients, and DECnet. This process ensured that the application would be properly serviced in the lights out environment.

In the benchmark phase, data was gathered from the application in order to make evaluations and decisions on load balancing, VAXcluster capacity, and network access. During testing, the application was run in a live environment to determine if additional modifications to the application, system, cluster, or environment would be required. During the acceptance and implementation planning phases, changes were communicated to the customers and the application was transitioned to production environment. Concurrent with this phase, DECalert tables were loaded with information regarding predetermined application decisions, notification routines, and responsible personnel.

Thus, a partnership was developed between the application development group and personnel responsible for planning, upgrading, and managing the lights out operating environment.

11.5.5 Overall Operations Support Implementation

In planning for a lights out computing environment, the CSC data center developed and implemented an operational support strategy encompassing the entire data center. The support strategy focused on:

- Management and implementation of established processes and procedures for application development and implementation.
- Management and staffing of a data center and centralized help line for system, application, facility, security, and critical event support.
- Enhancing and upgrading VMS, VAXcluster, and environmental operating systems with Digital proprietary software programs specific to a lights out operating environment.
- Remote fault isolation, resolution, and repair of hardware, system, and software products in addition to network components and telecommunication services.
- Remote fault isolation, resolution, and repair of electrical, HVAC, and data center support equipment.

These services required an experienced maintenance management organization with skills and expertise in maintenance planning, security, operational support delivery, and project management, in a multivendor, multitechnology environment.

One of the applications developed by the data center personnel that performed fault management tasks became a Digital product called DECalert. See Section 11.6 for details.

11.6 DECalert and Other Products Used to Manage CSC Data Center Operations

In order to ensure the dependability of CSC computing services and to improve the quality of their own personal lives, CSC data center personnel built an internal application that performed fault management tasks. The data center personnel had several goals in designing their system and selecting (or building their own) data center tools:

- To meet CSC's business requirement for 100% application availability.
- To keep operational costs down by minimizing the size of the staff necessary to manage the data center.
- To build sensors and alarms so that the computing systems can be monitored automatically. Over time, to enhance this high degree of fault management as new scenarios for potential problems are realized.
- To allow the remaining data center staff to enjoy their assignments by concentrating on implementing always changing data center technologies (as opposed to mounting backup magnetic tapes from 10 p.m. to 2 a.m.).

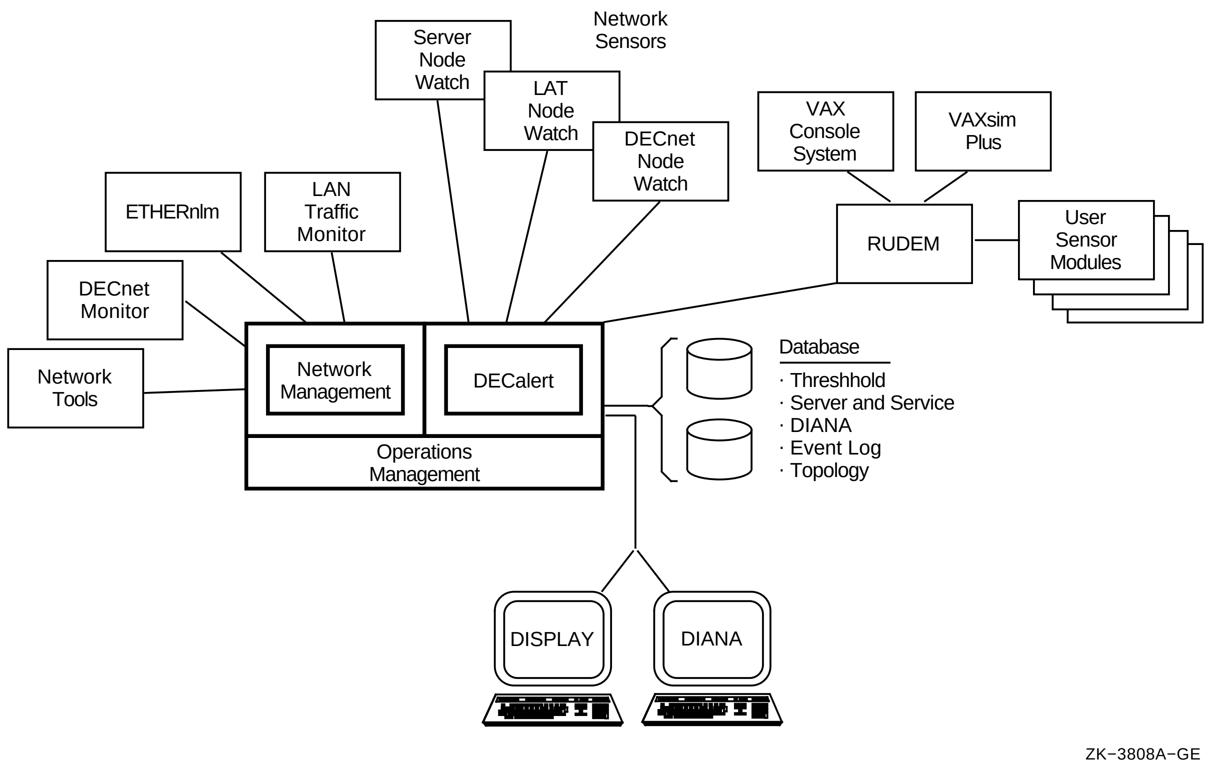
By 1991 the fault management alerting application had evolved into a Digital product called DECalert and the data center had developed a comprehensive set of customized sensors to detect potential problem situations.

Note

In February 1994, Digital granted Target Systems Corporation a nonexclusive, nontransferable license to use DECalert Version 1.2 software. Under the agreement, Target Systems may provide support for the Digital customer base (existing and future) for the product, as well as future development and migration of the product or its derivatives. Target Systems named the product TARGET–>ALERT.™ Digital reserves the right to continue to sell DECalert or its enhancements, as Digital deems fit.

Figure 11–1 illustrates DECalert and other products used to manage the CSC data center operations.

Figure 11–1 CSC Data Center Operations Management



As the central component, or *heart* of the CSC data center operations, DECalert helped the data center staff:

- To improve system availability by avoiding potential problems through early detection
- To save costs by allowing the data center to better use their personnel and to minimize the need for additional staff to support expanding computing systems at CSC

As an alerting technology for data centers, DECalert is designed to consolidate and distribute system and user events to the appropriate personnel. As a data center support application, DECalert monitors systems and network components such as DECnet Phase IV nodes, LAT service nodes, and terminal servers. DECalert provides:

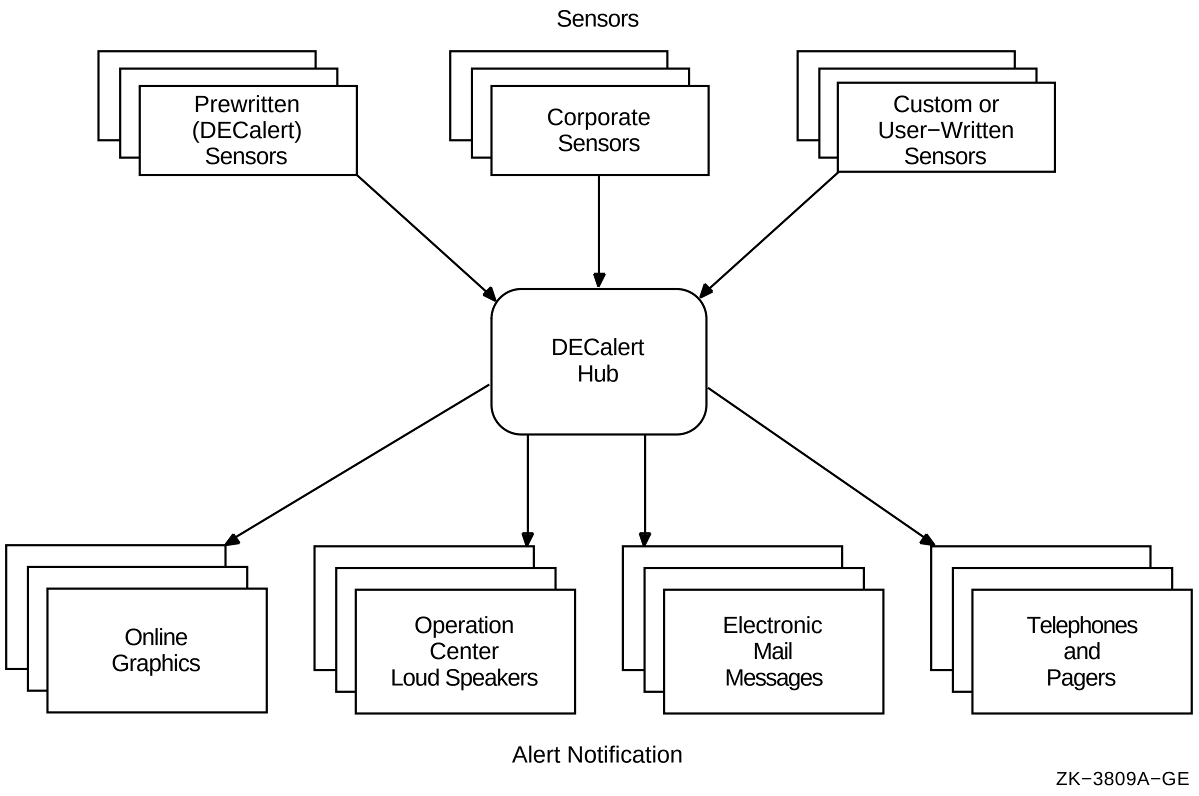
- Consolidation and distribution of data center alarms
- Support of custom-written sensors
- Self-monitoring features for high reliability
- Alerting options to set a required response or no required response
- Event logging and filtering
- Automatic escalation procedures if an alert that requires a response within a given time period is not being answered
- Report generation
- A playback feature that allows for review of all reported events

DECalert helps the CSC computing services team anticipate data center problems through early detection. One of the fundamental standards in data center management is fault prevention, which is the focal point of DECalert. DECalert uses four aspects of fault prevention:

- Detection
- Notification (main concentration of DECalert)
- Isolation
- Correction

Figure 11–2 depicts the DECalert sensors and alert notifications.

Figure 11–2 DECalert Sensors and Alert Notifications



In Figure 11–2:

- Prewritten DECalert sensors are DECalert product-specific sensor modules provided with the DECalert software kit. RUDEM, shown in Figure 11–1, is the module that enables you to write the custom-sensor applications.
- Corporate sensors are Digital software products that detect significant events. DECalert augments the event notification capability inherent in these products with additional alert notification features. The associated Digital products include Data Center Monitor (DCM), DEC Management Control Center (DECmcc), VAXsimPLUS, VMScluster Console System (VCS), and DECscheduler. These products are summarized later in this section.

- Custom-sensor applications are those that you can write to suit the needs of your systems and networks. You define the conditions and necessary alerting of personnel about potential problems. You can write the applications as DCL command procedures, programs, or both.

In addition, Figure 11–2 shows how DECalert performs event notification through:

- Online graphics

  A hierarchical display highlights all entities with outstanding alert conditions.
- Electronic mail messages

  The OpenVMS Mail Utility notifies designated personnel.
- Operation center loud speakers

  Loud speakers use synthesized DECtalk voice announcements that detail alert conditions.
- Telephone messages and pagers

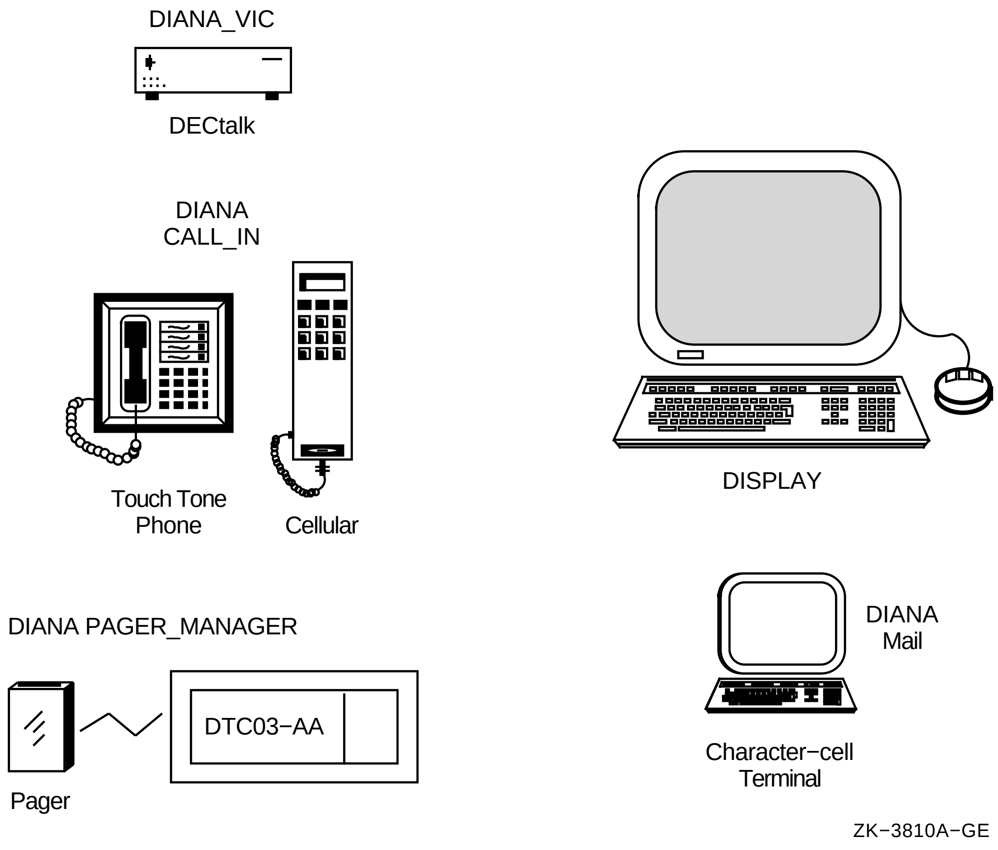
  Local or remote notifications can occur over telephones, voice pagers, and display pagers. DECalert provides support for paging systems that require pager identification numbers. Thus, DECalert can page using services such as SkyPager® and People Finder®.

The main DECalert processes, or *core software*, are:

- ALERT, which accepts alert messages from software sensors.
- DIANA, which performs all forms of notifications except for graphical displays. DIANA is an acronym for DECtalk Interface and Network Analysis.
- DISPLAY, which searches specific DECalert databases for information about the physical location of the device for which the alert message was generated.

Figure 11–3 shows the DECalert modules that perform notifications and displays.

Figure 11–3 DIANA Modules and DISPLAY



In Figure 11–3:

- DIANA_VIC, or DIANA Verbal Information Center, coordinates all voice announcements
- DIANA CALL_IN lets you call in to the DECalert system to retrieve messages posted with the pager
- DIANA Mail provides for VAX mail notifications of DECalert alert conditions
- DIANA PAGER_MANAGER communicates changes in the monitored environment to the support personnel by paging with displays, voice pagers, paging services, cellular telephones, and touch-tone telephones

Related products used to manage the CSC data center operations:

- NMCC/Ethernet Network Integrity Monitor (ETHERnim) is a network management tool that aids in fault isolation and configuration management of extended LANs. ETHERnim is an OpenVMS layered product that listens to a network and builds a database and graphic topology map of the extended LAN. ETHERnim recognizes Digital devices and other vendor devices on the LAN or extended LAN and includes them in the database and topology map. ETHERnim is designed to test the communications path through the Ethernet protocol and DECnet.

With ETHERnim, the network managers at the CSC data center can reference a large amount of information about each node via a database that can be edited. ETHERnim can determine the processor type for OpenVMS DECnet nodes on the Ethernet. Refer to the *Telecommunications and Networks Buyer's Guide* and the ETHERnim documentation set for details about ETHERnim.

- LAN Traffic Monitor (LTM) provides information needed to maximize network performance and plan network growth. LTM analyzes the use of an extended LAN. This product provides network use graphs that show the network manager the performance of the various LAN segments. LTM gathers traffic statistics from any device connected to the LAN and provides data based on nodes, addresses, and protocol types. Refer to the *Telecommunications and Networks Buyer's Guide* and the LAN Traffic Monitor documentation set for details about LTM.
- DECnet Monitor is a software tool for the observation and control of complex, corporate backbone networks, such as the one that exists between Digital engineering and support organizations, including the three U.S. CSC sites in Colorado, Georgia, and Massachusetts.

  DECnet Monitor responds to English-like commands and graphically presents network conditions on a color (or monochrome) network topology map. DECnet Monitor features include:

  – Nodes that can be instructed to send selected DECnet events to DECnet Monitor, helping (along with DECalert) to avoid major problems and minimize node down time because it enables a manager to detect problems early.

  – Relational database capabilitites that provide for active or passive collection of statistics.

  – Visual displays of network configuration and performance information that simplify interpretation of data and highlight conditions requiring attention. The network topology is displayed as a logical map for recognition of network elements and potential trouble areas.

  – Histograms and bar charts that display historical information to help data center staff analyze trends and plan for growth and change.

  – Real-time event logging with predetermined polling parameters that enable data center staff to detect problems early.
- Data Center Monitor (DCM) checks system and application processes and printer and batch queues, reporting abnormal events via a screen display, VAX electronic mail, or DECtalk. DCM eliminates the need for system support staff to continuously monitor operations of the data center by providing notification the moment an abnormal event occurs. DCM currently includes about 30 defined events. Three examples of DCM events that could be used by the CSC data center personnel:

  - A disk is approaching a defined threshold of remaining free blocks
  - A shadow set is losing a member disk
  - A queue is stopped

  DECalert interfaces with DCM and provides sophisticated notification technology. DCM provides built-in event coverage that is not available with DECalert (unless the user creates the event sensing code).

- DEC Management Control Center (DECmcc) is a multivendor network management platform that coordinates interactions among various network modules. With DECmcc, several network systems (including those from computer vendors other than Digital) can be monitored simultaneously.

  DECalert interfaces with the Digital Enterprise Management Architecture (EMA) compliant version of DECmcc via the DECmcc ALARMS functional module.

- The VMScluster Console System (VCS) helps the data center staff to consolidate system management functions by replacing multiple hardcopy console terminals with a single VCS. When VCS is linked to each device, all console messages are sent to a single console. VCS time-stamps all data received from each node and records it in a central disk file. This information can be retrieved at specific time intervals for display or printout by any terminal connected to the VCS.

  Using VCS reduced space constraints and cut overhead. Before VCS, system managers and operators looked for error messages (on reams of paper near each node) about conditions that might adversely affect system performance and reliability. The benefits of VCS include:

  - Better console and system management
  - Improved operator and manager productivity
  - Increased floor space in labs (where hardcopy consoles formerly resided at each VAX node)
  - Reduced clutter, noise, and maintenance costs
  - Improved ability to plan changes to the installation configurations

  The CSC data center staff uses VCS to capture system error messages on a centralized console. The CSC data center personnel uses VCS for its console consolidation capability and to handle events that are reported via the console. When possible, events are forwarded to DECalert.

  As noted elsewhere in this handbook, part of achieving system availability and dependability is fast recovery. VCS console consolidation lets an operator connect to the console ports of (up to) 32 systems (AXP nodes, VAX nodes, HSC devices, others) from a single workstation. This feature enables an operator to reboot a system from the control room while still having access to other information being reported by DECalert, DCM, and other products.

- VAXsimPLUS works with the RUDEM component of DECalert to reduce down time by continuously monitoring system performance. VAXsimPLUS identifies intermittent problems, predicts failures, and (in the case of RA* disks) initiates corrective action.

  With RA™ Winchester disks, the early warning capability of VAXsimPLUS enables its autocopy feature to automatically invoke Volume Shadowing for OpenVMS when a drive is in the warning condition. VAXsimPLUS identifies the problem while making a backup copy of the suspect drive onto the spare drive. This dynamic disk substitution ensures that data availability and integrity are maintained. Users continue to run their applications from the spare drive while the original drive is taken off line and serviced. Once the drive has been repaired, the data is restored to the original drive, returning the backup drive to its original shadow set for future autocopy needs.

- The data center uses VAX Storage Library System (SLS) to manage their removable tape media. SLS manages most VMS-supported removable-media storage devices include reel-to-reel magnetic tape drives, cartridge tape drives, the RV20 optical drive, and the RV64 optical tape drive. The data center uses SLS primarily to support the automated TA90 cartridge loader, select a group of scratch tapes, and to place them in the loader for unattended backups. As each tape is loaded, SLS verifies that it is a scratch tape (and not a data tape loaded by mistake) before writing on it. SLS reports inconsistencies with tape labels to the operator. SLS can also track media movement between data center and off-site storage and notifies system users if media are unavailable or in remote storage.

  In general, SLS features also allow operators to read and write both ANSI and IBM tapes. It translates EBCDIC (IBM) to ASCII (most other vendors) code and vice versa. This capability lets you use nine-track or TA90 cartridge tapes to move data back and forth between Digital and IBM systems. SLS can also validate, read, and write ANSI tapes. The software can read (but not validate) IBM labels, and it can write unlabeled IBM tapes.

11.7 Additional Benefits of the Lights Out Environment

The transition to a lights out computing strategy provided CSC with a significant opportunity to realize additional organizational benefits. They included:

- Reducing long term operating costs
- Moving directly to an advanced high performance organization
- Molding contingencies directly into strategies
- Increasing security measures, controls, and awareness
- Providing optimum advantages in state-of-the-art technology
- Increasing flexibility
- Developing proactive data center management
- Increasing awareness of their business

A
Data Center Evaluation Checklists

This appendix contains data center evaluation checklists that you can use to determine whether your current computing resources meet the minimum suggested requirements for a dependable computing environment. The checklists are useful for identifying the elements that you must manage to achieve a dependable computing environment.

The questions are structured so that the *correct* answer is *Yes*. If, after completing the checklists, you find that you have answered a majority of the questions with a *No,* you should take steps to remedy the particular inadequacy. If you need professional help, contact your local Digital support representative or call 1-800-DIGITAL (1-800-344-4825) for inquiries about consulting services; a Digital consultant can help you analyze areas that need improvement and define a schedule for implementing changes to create a more dependable computing environment.

Note

The checklists are available on line in SYS$EXAMPLES[1] Print VMS_DEPENDABILITY_CHECKLIST.PS if you have a PostScript printer and wish to complete one or more copies of the checklist in paper form. Print VMS_DEPENDABILITY_CHECKLIST.TXT if you do not have a PostScript printer. A third option is to copy VMS_DEPENDABILITY_CHECKLIST.TXT from SYS$EXAMPLES to a private directory, invoke a text editor, and complete the checklist on line.

---

[1] VAX systems only; the information in the checklists is relevant, however, for data centers using AXP systems, VAX systems, or both.

The evaluation checklists are organized into the following categories:

- General planning (see Section A.1.)

  Comprehensive, well-documented plans are essential to a data center. The general planning checklist evaluates your data center's general planning strategy.
- Environment management (see Section A.2.)

  Effective management of your data center's management is critical. The environment management checklist evaluates your data center's environmental management strategy.
- Data center organization (see Section A.3.)

  The data center organization checklist evaluates your overall strategy.
- Security (see Section A.4.)

  Strict security strategies must be implemented to protect your data center from intruders. The security checklist evaluates your data center's security strategy.

  Caution

  Use discretion in selecting which options to enable. Extensive use of security auditing can consume significant system resources.

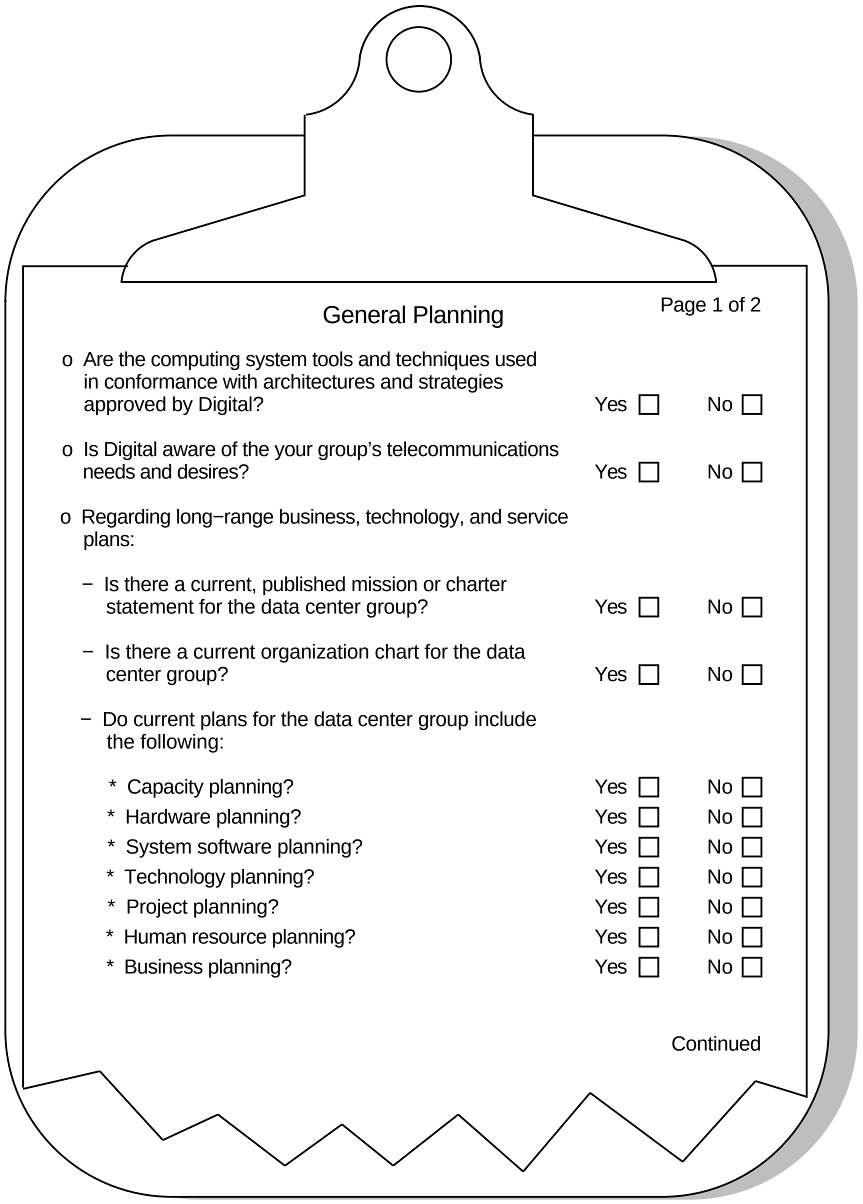
- Application software (see Section A.5.)

  Vendor-supplied application software and your group's application software must be evaluated and tested extensively before it can qualify to be installed on production systems in your data center. The application software checklist evaluates your data center's application software strategy.
- Digital service and support (see Section A.6.)

  Digital specialists can help you plan strategies for building a data center. The Digital service and support checklist evaluates your data center's support strategy.

Section A.7 contains a compliance summary table so that you can determine how well your data center complies with suggested techniques for achieving a dependable computing environment.

A.1 General Planning Checklist



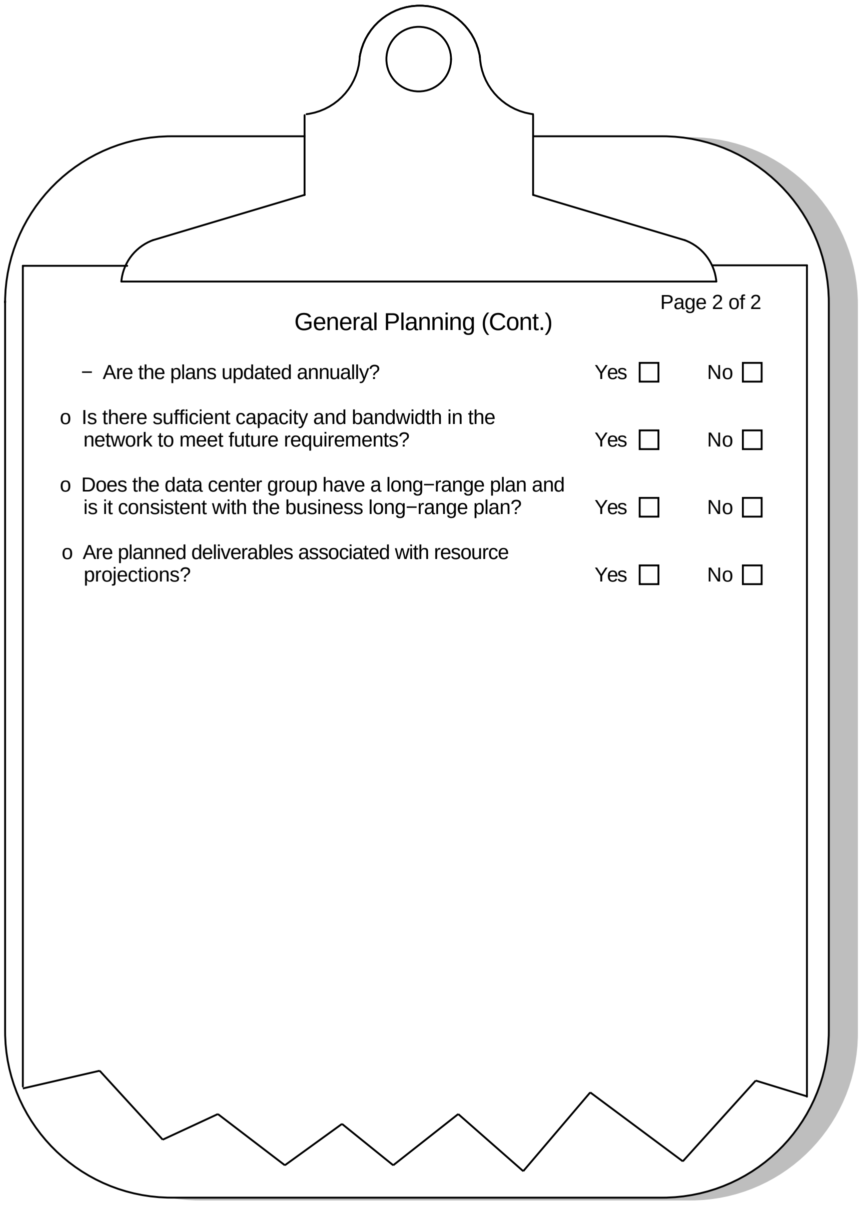
General Planning

Page 1 of 2

o Are the computing system tools and techniques used in conformance with architectures and strategies approved by Digital?	Yes ☐	No ☐
o Is Digital aware of the your group's telecommunications needs and desires?	Yes ☐	No ☐
o Regarding long−range business, technology, and service plans:		
− Is there a current, published mission or charter statement for the data center group?	Yes ☐	No ☐
− Is there a current organization chart for the data center group?	Yes ☐	No ☐
− Do current plans for the data center group include the following:		
* Capacity planning?	Yes ☐	No ☐
* Hardware planning?	Yes ☐	No ☐
* System software planning?	Yes ☐	No ☐
* Technology planning?	Yes ☐	No ☐
* Project planning?	Yes ☐	No ☐
* Human resource planning?	Yes ☐	No ☐
* Business planning?	Yes ☐	No ☐

Continued

ZK−3835A−GE



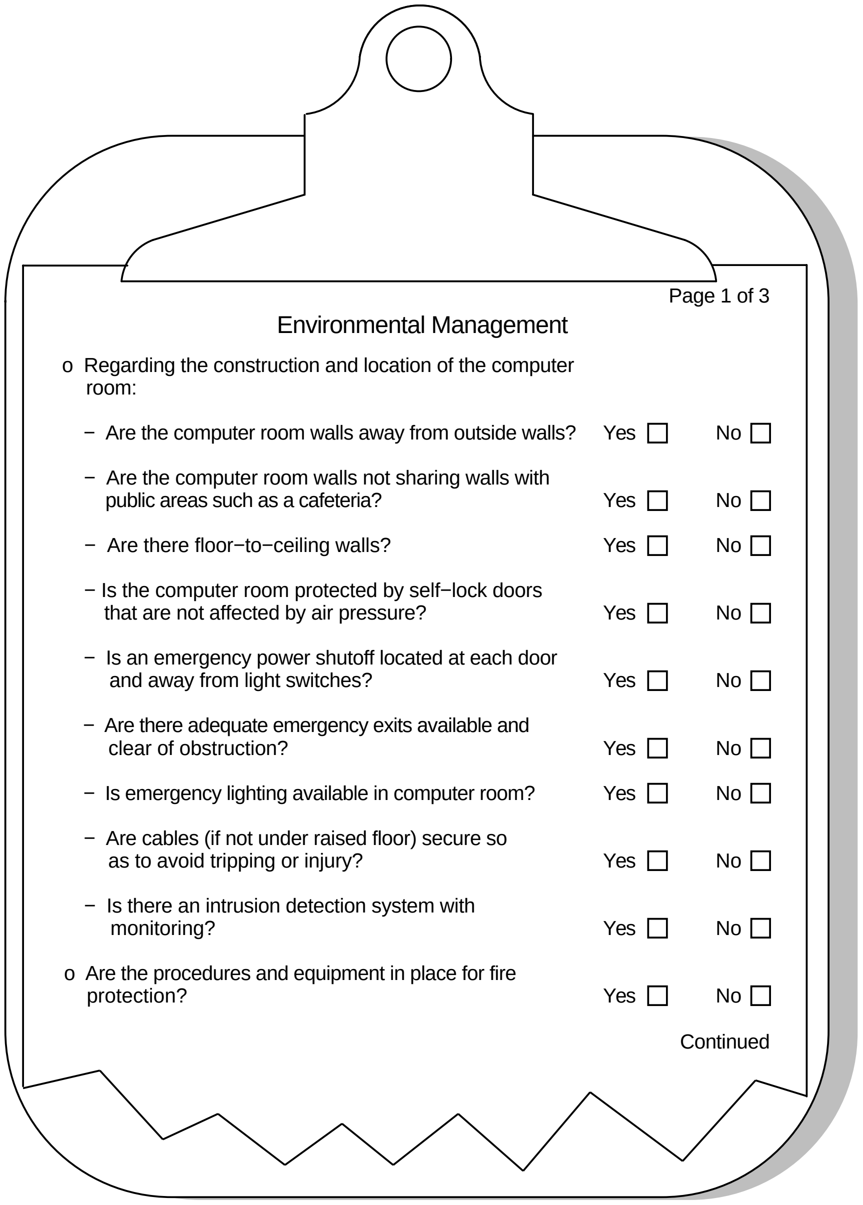
Page 2 of 2

General Planning (Cont.)

− Are the plans updated annually?	Yes ☐	No ☐
o Is there sufficient capacity and bandwidth in the network to meet future requirements?	Yes ☐	No ☐
o Does the data center group have a long−range plan and is it consistent with the business long−range plan?	Yes ☐	No ☐
o Are planned deliverables associated with resource projections?	Yes ☐	No ☐

ZK−3836A−GE

A.2 Environmental Management Checklist



Page 1 of 3

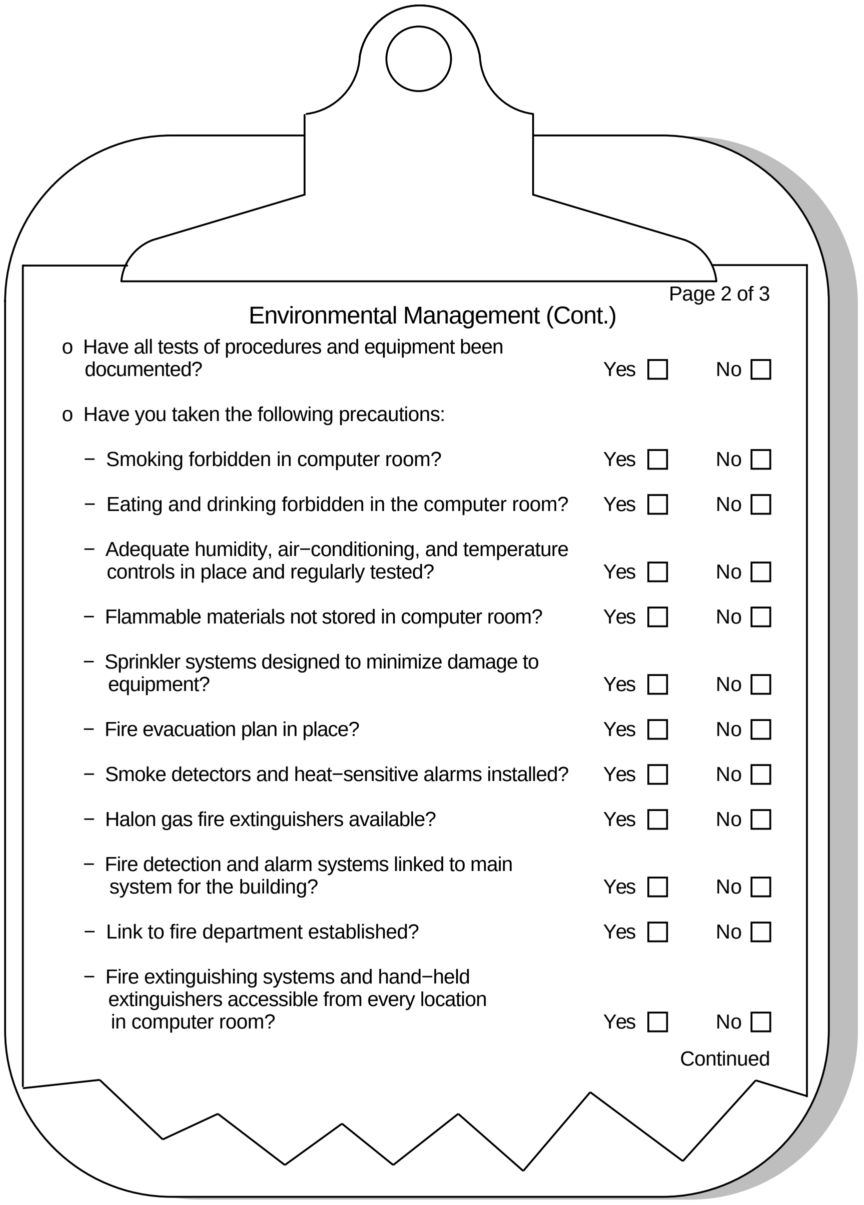
Environmental Management

o Regarding the construction and location of the computer room:

− Are the computer room walls away from outside walls?	Yes ☐	No ☐
− Are the computer room walls not sharing walls with public areas such as a cafeteria?	Yes ☐	No ☐
− Are there floor−to−ceiling walls?	Yes ☐	No ☐
− Is the computer room protected by self−lock doors that are not affected by air pressure?	Yes ☐	No ☐
− Is an emergency power shutoff located at each door and away from light switches?	Yes ☐	No ☐
− Are there adequate emergency exits available and clear of obstruction?	Yes ☐	No ☐
− Is emergency lighting available in computer room?	Yes ☐	No ☐
− Are cables (if not under raised floor) secure so as to avoid tripping or injury?	Yes ☐	No ☐
− Is there an intrusion detection system with monitoring?	Yes ☐	No ☐
o Are the procedures and equipment in place for fire protection?	Yes ☐	No ☐

Continued

ZK−3837A−GE



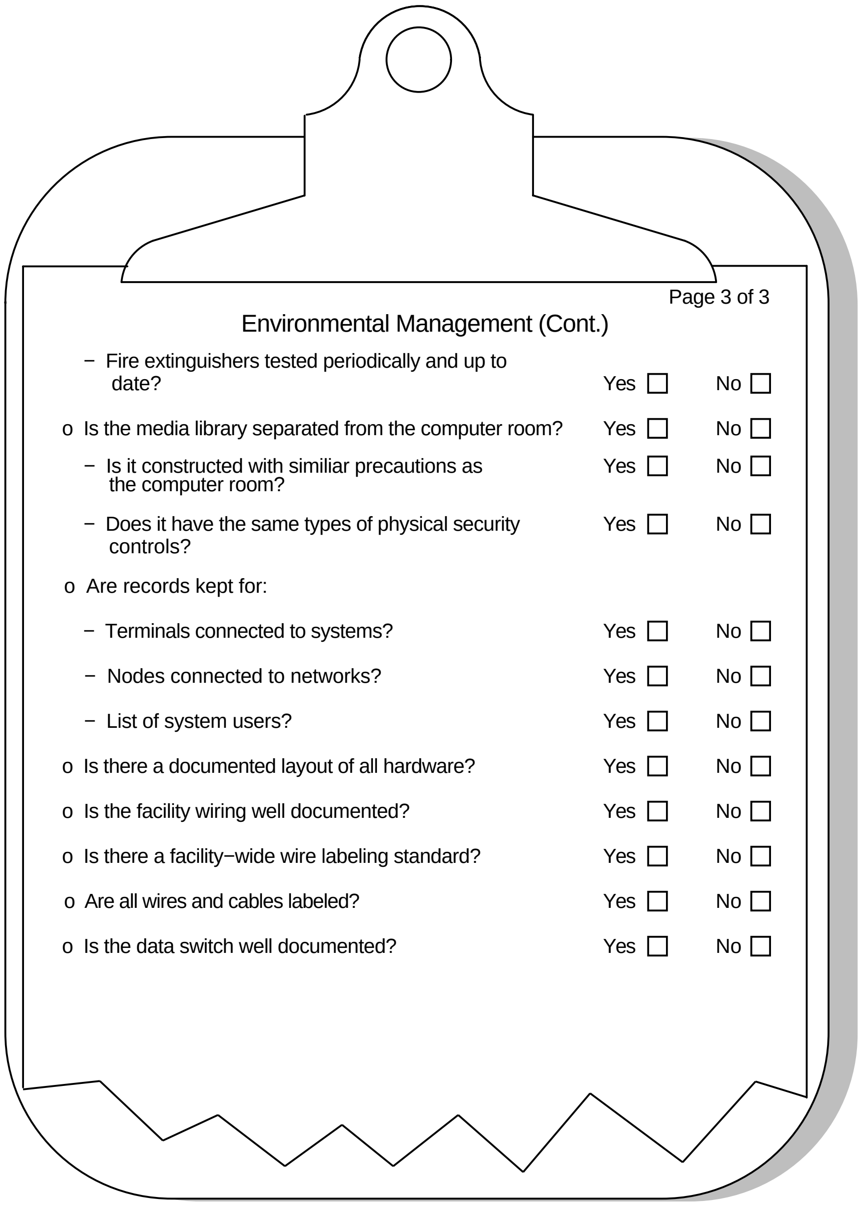
Page 2 of 3

Environmental Management (Cont.)

o Have all tests of procedures and equipment been documented?	Yes ☐	No ☐
o Have you taken the following precautions:		
− Smoking forbidden in computer room?	Yes ☐	No ☐
− Eating and drinking forbidden in the computer room?	Yes ☐	No ☐
− Adequate humidity, air−conditioning, and temperature controls in place and regularly tested?	Yes ☐	No ☐
− Flammable materials not stored in computer room?	Yes ☐	No ☐
− Sprinkler systems designed to minimize damage to equipment?	Yes ☐	No ☐
− Fire evacuation plan in place?	Yes ☐	No ☐
− Smoke detectors and heat−sensitive alarms installed?	Yes ☐	No ☐
− Halon gas fire extinguishers available?	Yes ☐	No ☐
− Fire detection and alarm systems linked to main system for the building?	Yes ☐	No ☐
− Link to fire department established?	Yes ☐	No ☐
− Fire extinguishing systems and hand−held extinguishers accessible from every location in computer room?	Yes ☐	No ☐

Continued

ZK−3838A−GE



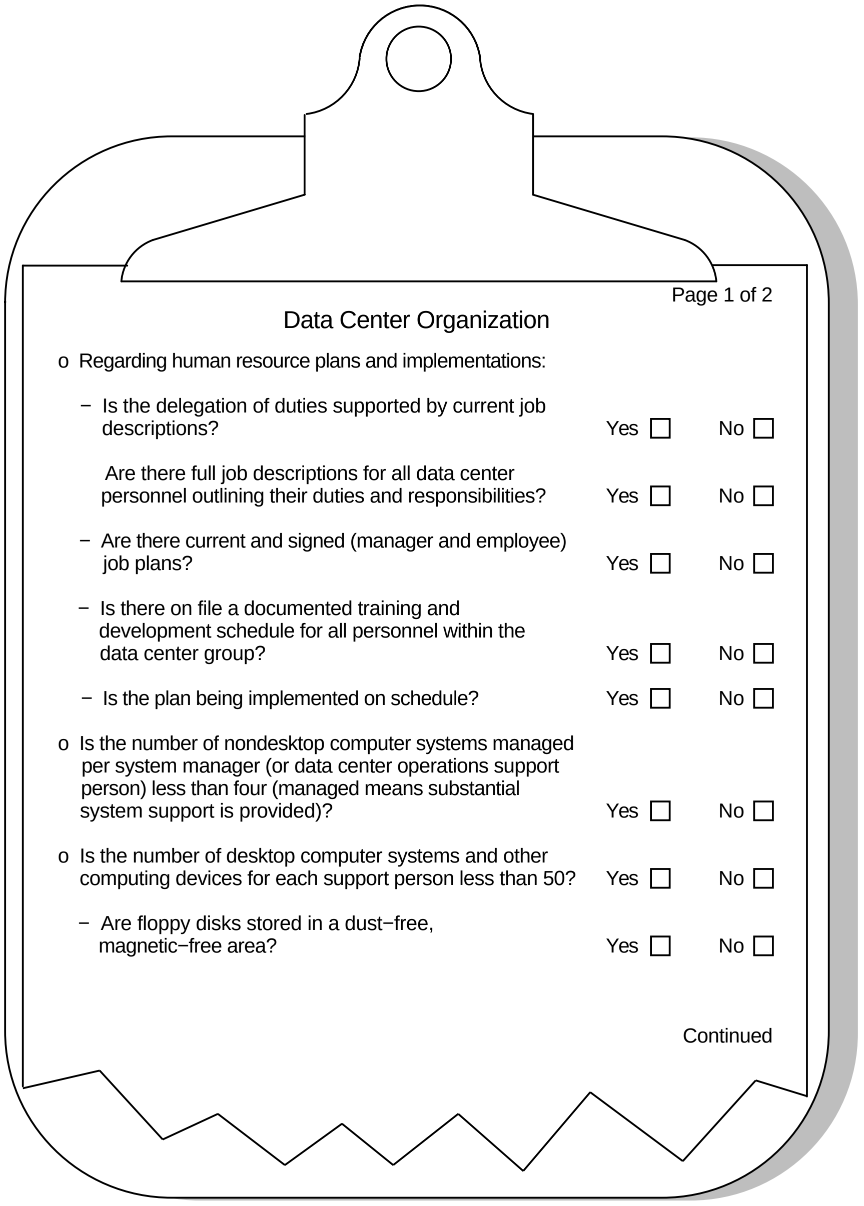
Page 3 of 3

Environmental Management (Cont.)

− Fire extinguishers tested periodically and up to date?	Yes ☐	No ☐
o Is the media library separated from the computer room?	Yes ☐	No ☐
− Is it constructed with similiar precautions as the computer room?	Yes ☐	No ☐
− Does it have the same types of physical security controls?	Yes ☐	No ☐
o Are records kept for:		
− Terminals connected to systems?	Yes ☐	No ☐
− Nodes connected to networks?	Yes ☐	No ☐
− List of system users?	Yes ☐	No ☐
o Is there a documented layout of all hardware?	Yes ☐	No ☐
o Is the facility wiring well documented?	Yes ☐	No ☐
o Is there a facility−wide wire labeling standard?	Yes ☐	No ☐
o Are all wires and cables labeled?	Yes ☐	No ☐
o Is the data switch well documented?	Yes ☐	No ☐

ZK−3839A−GE

A.3 Data Center Organization Checklist



Page 1 of 2

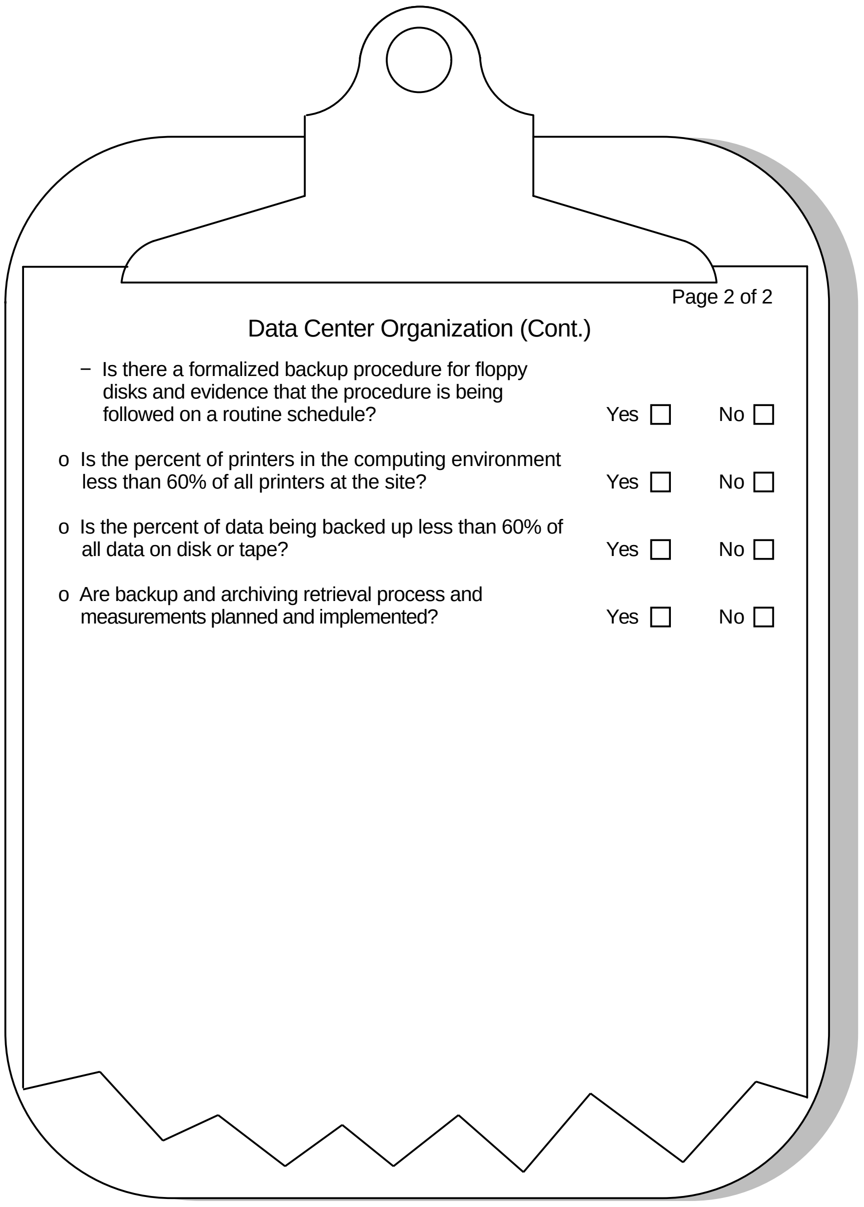
Data Center Organization

o Regarding human resource plans and implementations:

− Is the delegation of duties supported by current job descriptions?	Yes ☐	No ☐
Are there full job descriptions for all data center personnel outlining their duties and responsibilities?	Yes ☐	No ☐
− Are there current and signed (manager and employee) job plans?	Yes ☐	No ☐
− Is there on file a documented training and development schedule for all personnel within the data center group?	Yes ☐	No ☐
− Is the plan being implemented on schedule?	Yes ☐	No ☐
o Is the number of nondesktop computer systems managed per system manager (or data center operations support person) less than four (managed means substantial system support is provided)?	Yes ☐	No ☐
o Is the number of desktop computer systems and other computing devices for each support person less than 50?	Yes ☐	No ☐
− Are floppy disks stored in a dust−free, magnetic−free area?	Yes ☐	No ☐

Continued

ZK−3840A−GE



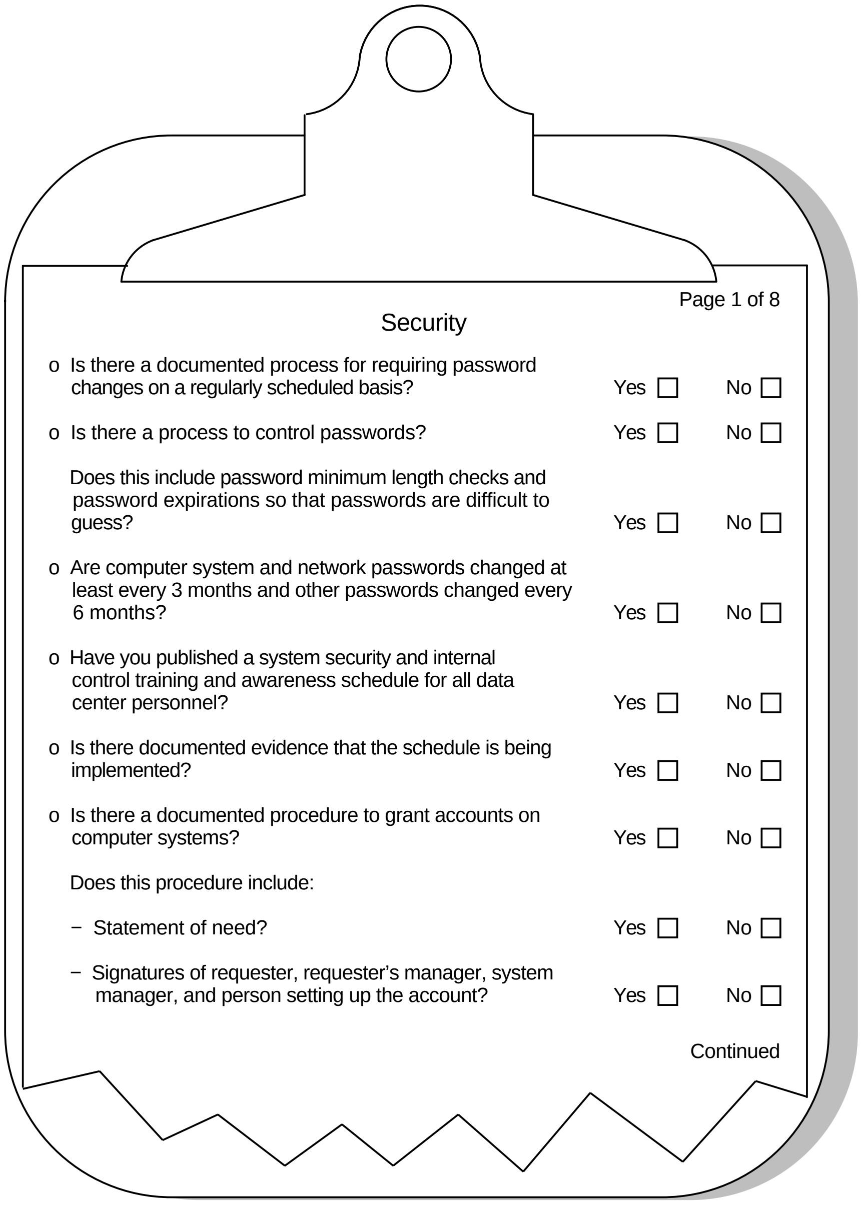
Page 2 of 2

Data Center Organization (Cont.)

− Is there a formalized backup procedure for floppy disks and evidence that the procedure is being followed on a routine schedule?	Yes ☐	No ☐
o Is the percent of printers in the computing environment less than 60% of all printers at the site?	Yes ☐	No ☐
o Is the percent of data being backed up less than 60% of all data on disk or tape?	Yes ☐	No ☐
o Are backup and archiving retrieval process and measurements planned and implemented?	Yes ☐	No ☐

ZK−3841A−GE

A.4 Security Checklist



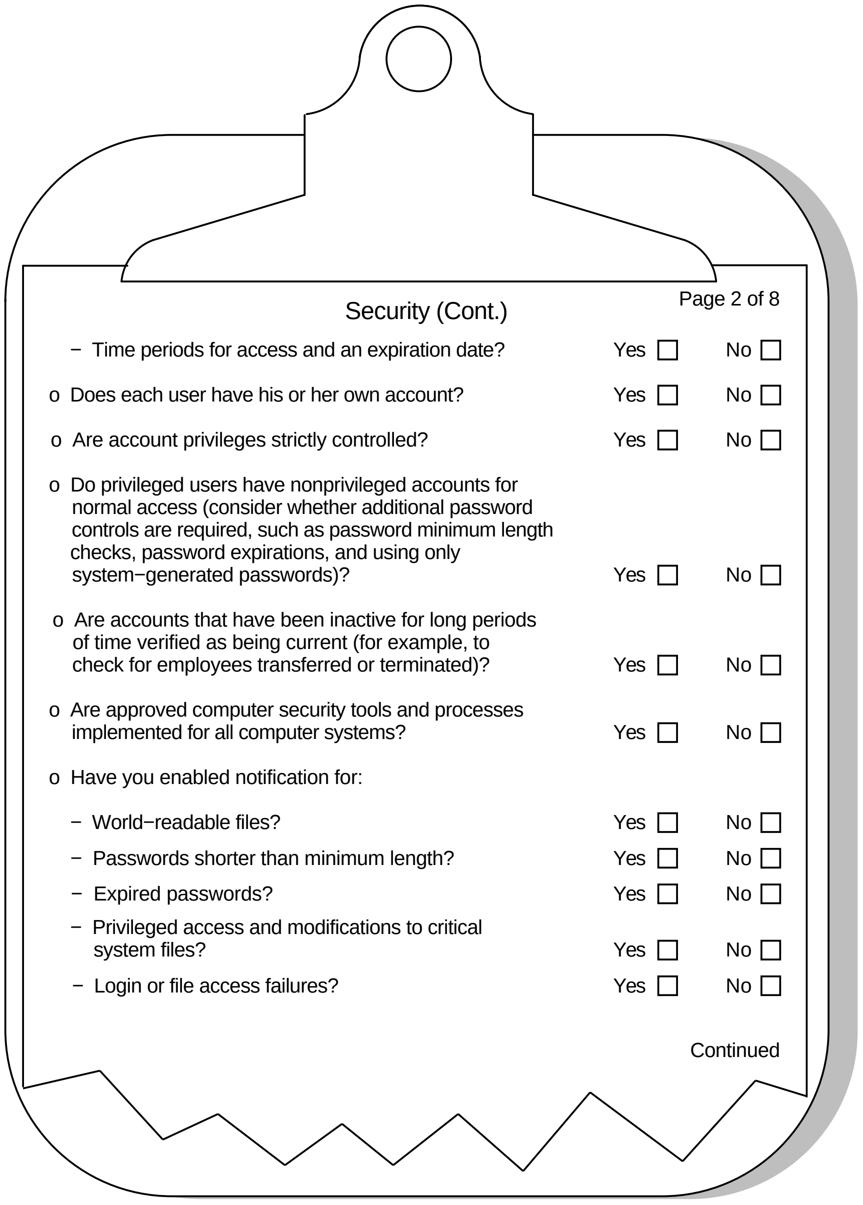
Page 1 of 8

Security

o Is there a documented process for requiring password changes on a regularly scheduled basis?	Yes ☐	No ☐
o Is there a process to control passwords?	Yes ☐	No ☐
Does this include password minimum length checks and password expirations so that passwords are difficult to guess?	Yes ☐	No ☐
o Are computer system and network passwords changed at least every 3 months and other passwords changed every 6 months?	Yes ☐	No ☐
o Have you published a system security and internal control training and awareness schedule for all data center personnel?	Yes ☐	No ☐
o Is there documented evidence that the schedule is being implemented?	Yes ☐	No ☐
o Is there a documented procedure to grant accounts on computer systems?	Yes ☐	No ☐
Does this procedure include:		
− Statement of need?	Yes ☐	No ☐
− Signatures of requester, requester's manager, system manager, and person setting up the account?	Yes ☐	No ☐

Continued

ZK−3842A−GE



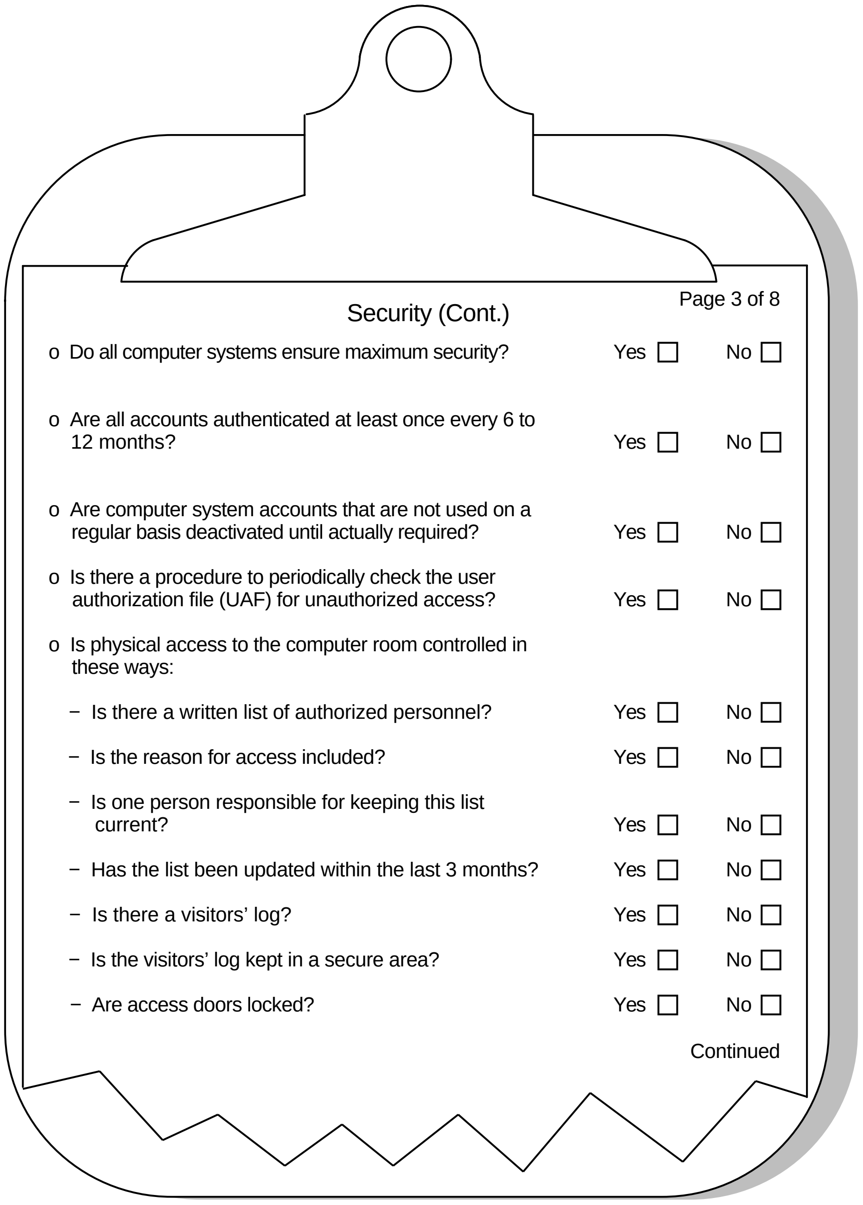
Page 2 of 8

Security (Cont.)

− Time periods for access and an expiration date?	Yes ☐	No ☐
o Does each user have his or her own account?	Yes ☐	No ☐
o Are account privileges strictly controlled?	Yes ☐	No ☐
o Do privileged users have nonprivileged accounts for normal access (consider whether additional password controls are required, such as password minimum length checks, password expirations, and using only system−generated passwords)?	Yes ☐	No ☐
o Are accounts that have been inactive for long periods of time verified as being current (for example, to check for employees transferred or terminated)?	Yes ☐	No ☐
o Are approved computer security tools and processes implemented for all computer systems?	Yes ☐	No ☐
o Have you enabled notification for:		
− World−readable files?	Yes ☐	No ☐
− Passwords shorter than minimum length?	Yes ☐	No ☐
− Expired passwords?	Yes ☐	No ☐
− Privileged access and modifications to critical system files?	Yes ☐	No ☐
− Login or file access failures?	Yes ☐	No ☐

Continued

ZK−3843A−GE



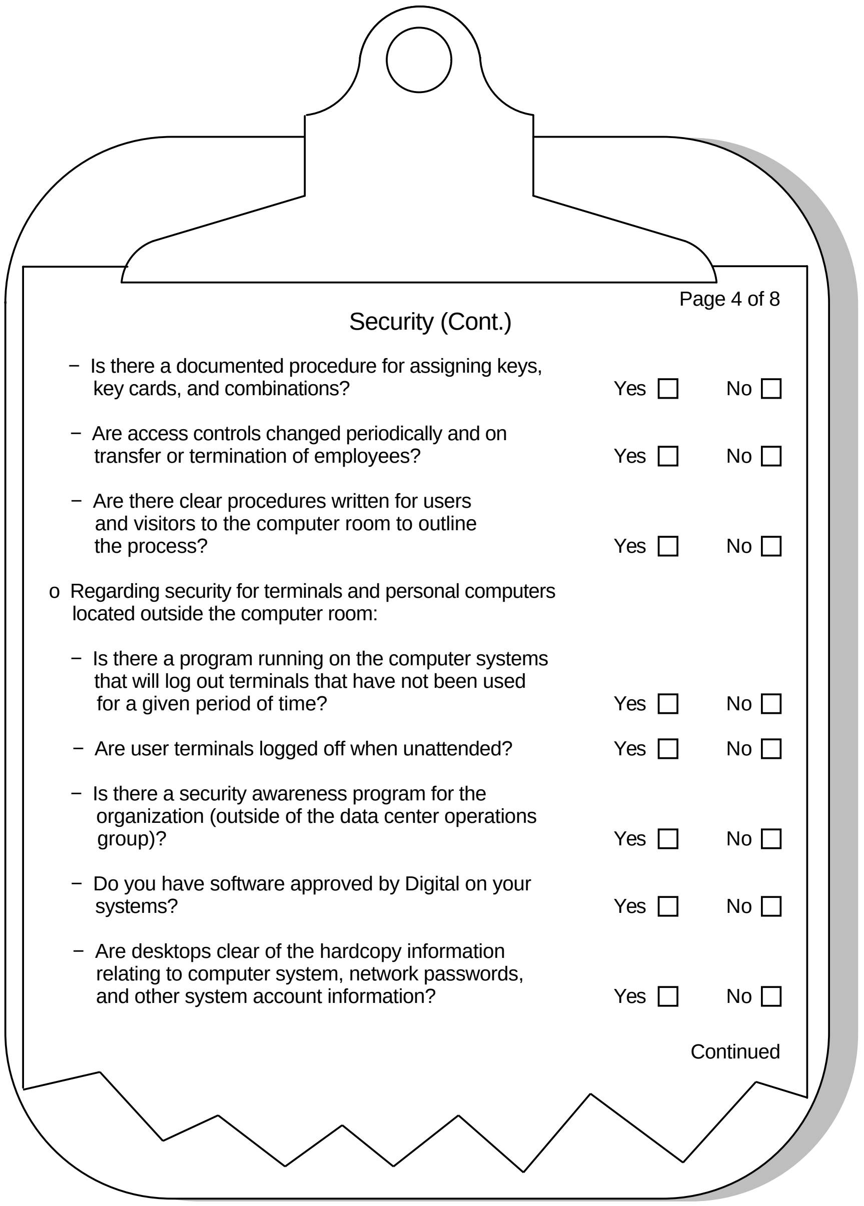
Page 3 of 8

Security (Cont.)

o Do all computer systems ensure maximum security?	Yes ☐	No ☐
o Are all accounts authenticated at least once every 6 to 12 months?	Yes ☐	No ☐
o Are computer system accounts that are not used on a regular basis deactivated until actually required?	Yes ☐	No ☐
o Is there a procedure to periodically check the user authorization file (UAF) for unauthorized access?	Yes ☐	No ☐
o Is physical access to the computer room controlled in these ways:		
− Is there a written list of authorized personnel?	Yes ☐	No ☐
− Is the reason for access included?	Yes ☐	No ☐
− Is one person responsible for keeping this list current?	Yes ☐	No ☐
− Has the list been updated within the last 3 months?	Yes ☐	No ☐
− Is there a visitors' log?	Yes ☐	No ☐
− Is the visitors' log kept in a secure area?	Yes ☐	No ☐
− Are access doors locked?	Yes ☐	No ☐

Continued

ZK−3844A−GE



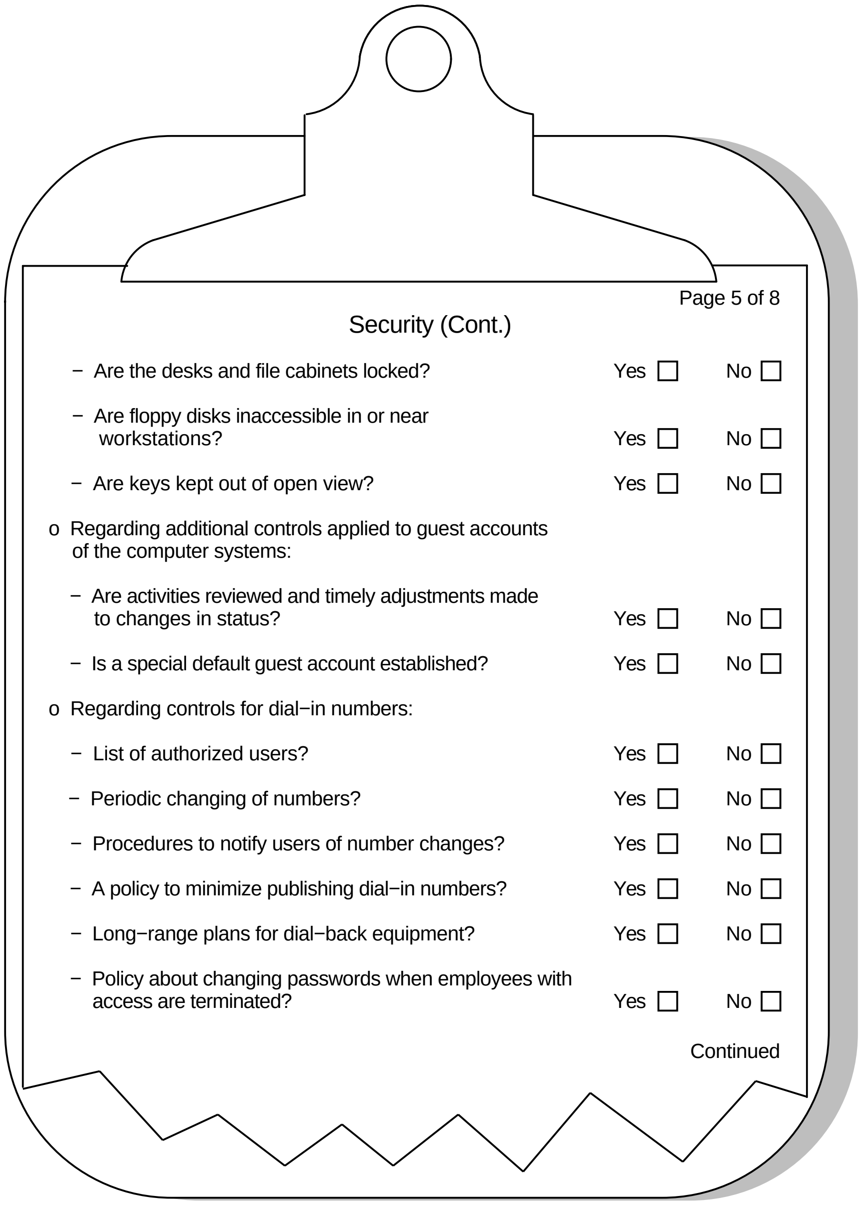
Page 4 of 8

Security (Cont.)

− Is there a documented procedure for assigning keys, key cards, and combinations?	Yes ☐	No ☐
− Are access controls changed periodically and on transfer or termination of employees?	Yes ☐	No ☐
− Are there clear procedures written for users and visitors to the computer room to outline the process?	Yes ☐	No ☐
o Regarding security for terminals and personal computers located outside the computer room:		
− Is there a program running on the computer systems that will log out terminals that have not been used for a given period of time?	Yes ☐	No ☐
− Are user terminals logged off when unattended?	Yes ☐	No ☐
− Is there a security awareness program for the organization (outside of the data center operations group)?	Yes ☐	No ☐
− Do you have software approved by Digital on your systems?	Yes ☐	No ☐
− Are desktops clear of the hardcopy information relating to computer system, network passwords, and other system account information?	Yes ☐	No ☐

Continued

ZK−3845A−GE



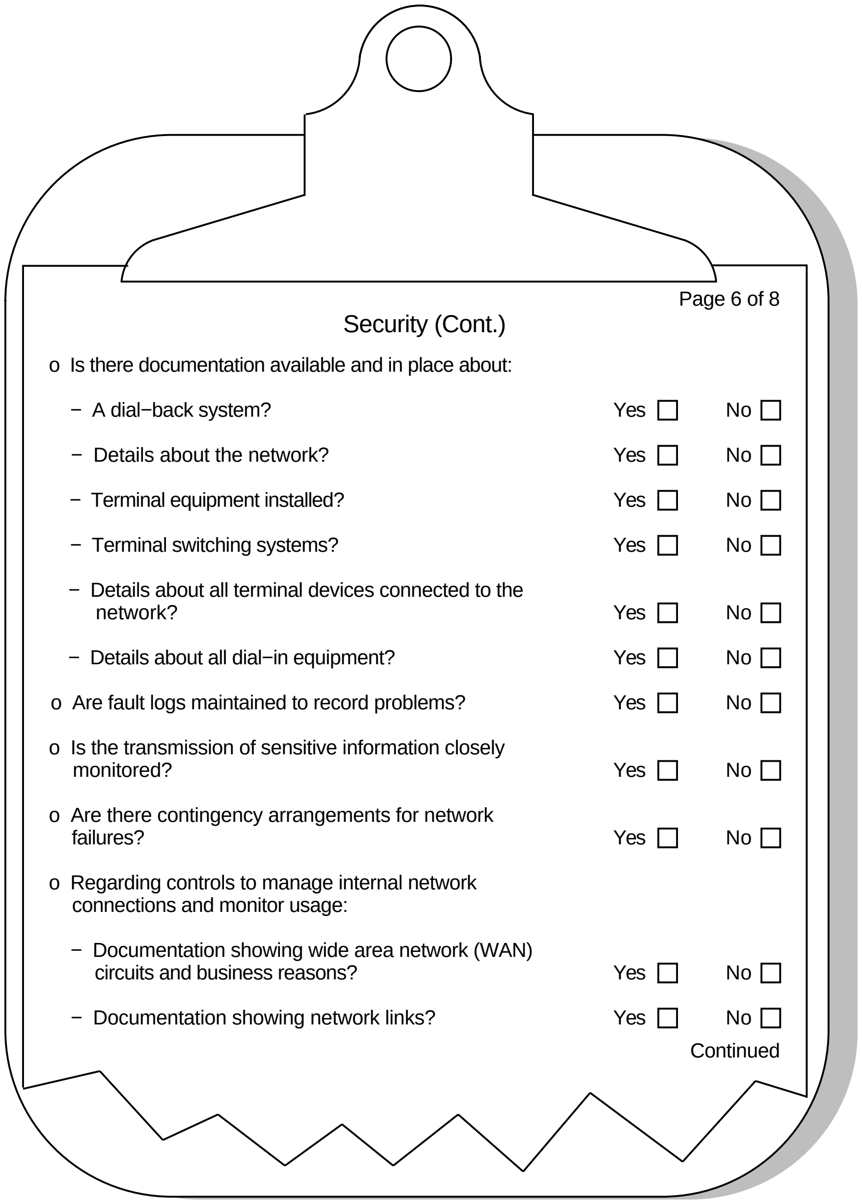
Page 5 of 8

Security (Cont.)

− Are the desks and file cabinets locked?	Yes ☐	No ☐
− Are floppy disks inaccessible in or near workstations?	Yes ☐	No ☐
− Are keys kept out of open view?	Yes ☐	No ☐
o Regarding additional controls applied to guest accounts of the computer systems:		
− Are activities reviewed and timely adjustments made to changes in status?	Yes ☐	No ☐
− Is a special default guest account established?	Yes ☐	No ☐
o Regarding controls for dial−in numbers:		
− List of authorized users?	Yes ☐	No ☐
− Periodic changing of numbers?	Yes ☐	No ☐
− Procedures to notify users of number changes?	Yes ☐	No ☐
− A policy to minimize publishing dial−in numbers?	Yes ☐	No ☐
− Long−range plans for dial−back equipment?	Yes ☐	No ☐
− Policy about changing passwords when employees with access are terminated?	Yes ☐	No ☐

Continued

ZK−3846A−GE



Page 6 of 8

Security (Cont.)

o Is there documentation available and in place about:

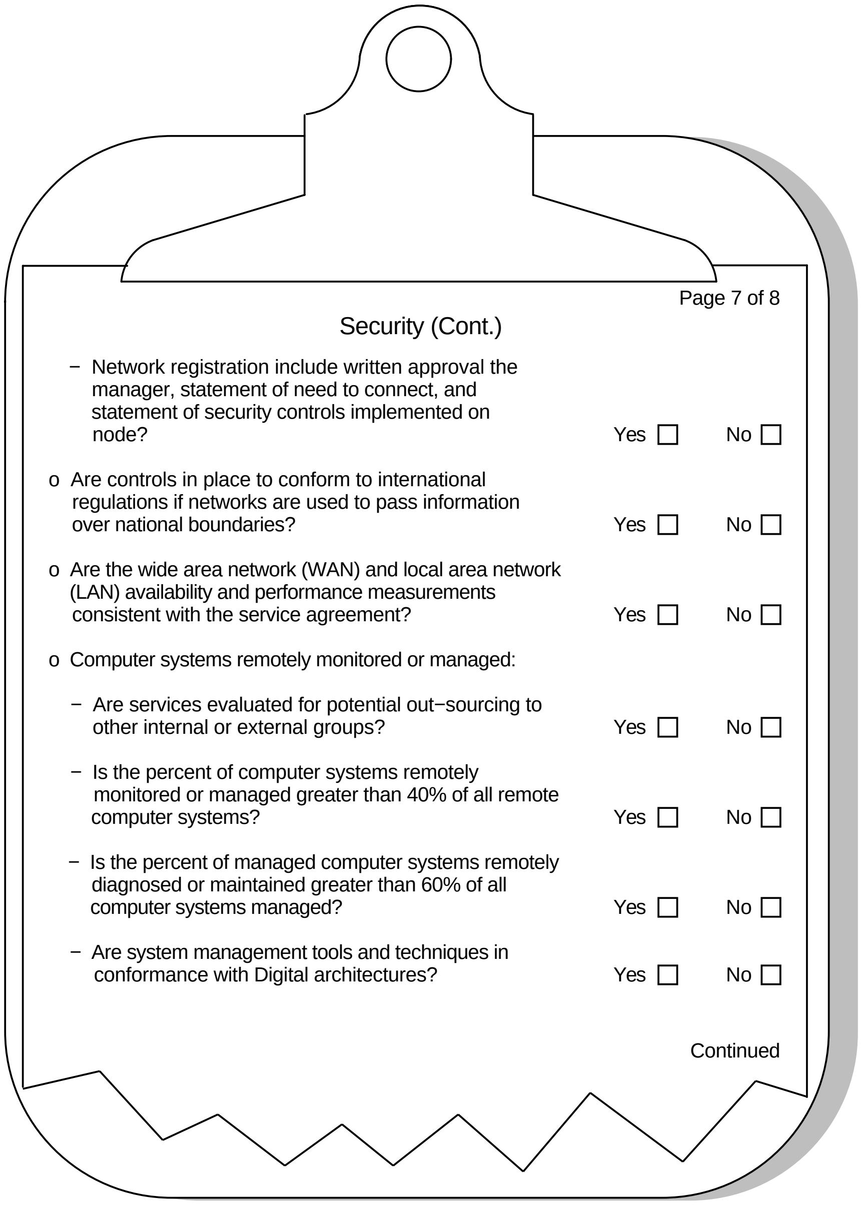
− A dial−back system?	Yes ☐	No ☐
− Details about the network?	Yes ☐	No ☐
− Terminal equipment installed?	Yes ☐	No ☐
− Terminal switching systems?	Yes ☐	No ☐
− Details about all terminal devices connected to the network?	Yes ☐	No ☐
− Details about all dial−in equipment?	Yes ☐	No ☐
o Are fault logs maintained to record problems?	Yes ☐	No ☐
o Is the transmission of sensitive information closely monitored?	Yes ☐	No ☐
o Are there contingency arrangements for network failures?	Yes ☐	No ☐

o Regarding controls to manage internal network connections and monitor usage:

− Documentation showing wide area network (WAN) circuits and business reasons?	Yes ☐	No ☐
− Documentation showing network links?	Yes ☐	No ☐

Continued

ZK−3847A−GE



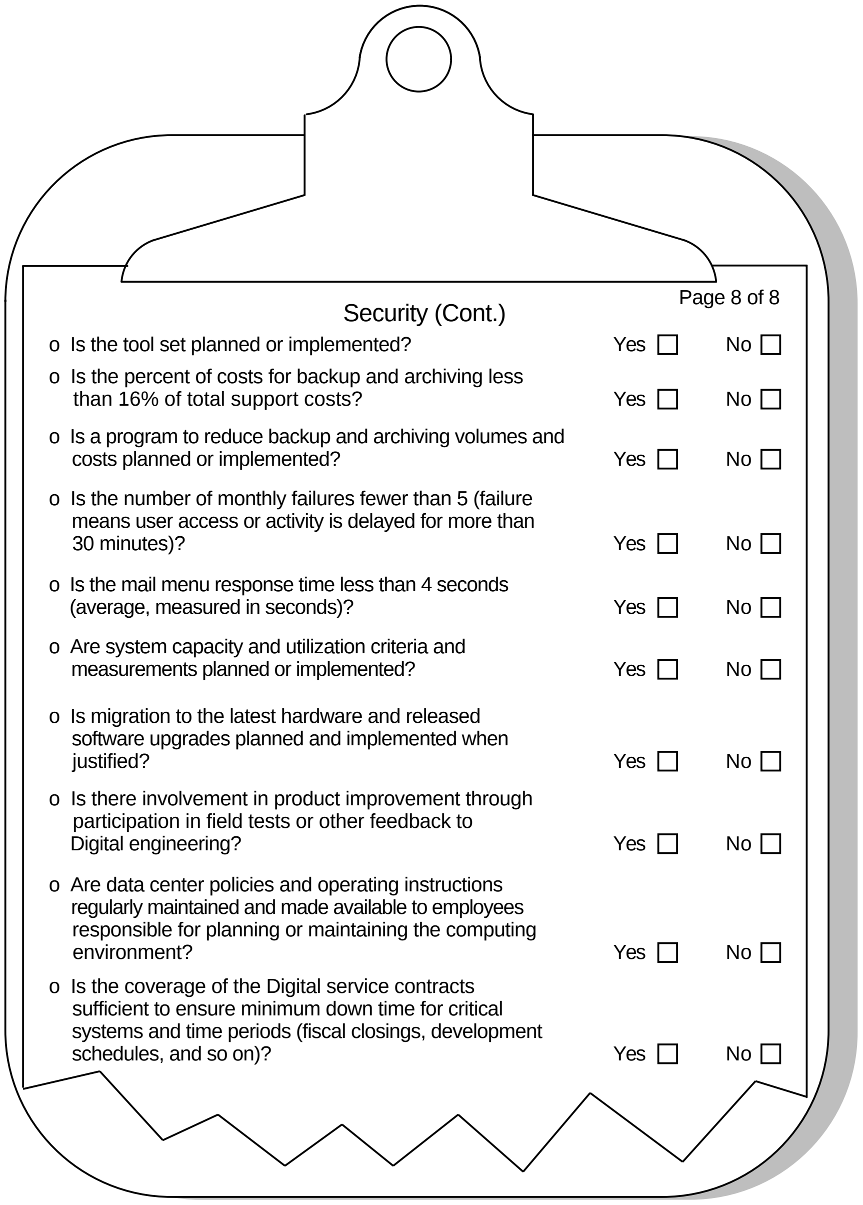
Page 7 of 8

Security (Cont.)

− Network registration include written approval the manager, statement of need to connect, and statement of security controls implemented on node?	Yes ☐	No ☐
o Are controls in place to conform to international regulations if networks are used to pass information over national boundaries?	Yes ☐	No ☐
o Are the wide area network (WAN) and local area network (LAN) availability and performance measurements consistent with the service agreement?	Yes ☐	No ☐
o Computer systems remotely monitored or managed:		
− Are services evaluated for potential out−sourcing to other internal or external groups?	Yes ☐	No ☐
− Is the percent of computer systems remotely monitored or managed greater than 40% of all remote computer systems?	Yes ☐	No ☐
− Is the percent of managed computer systems remotely diagnosed or maintained greater than 60% of all computer systems managed?	Yes ☐	No ☐
− Are system management tools and techniques in conformance with Digital architectures?	Yes ☐	No ☐

Continued

ZK−3848A−GE



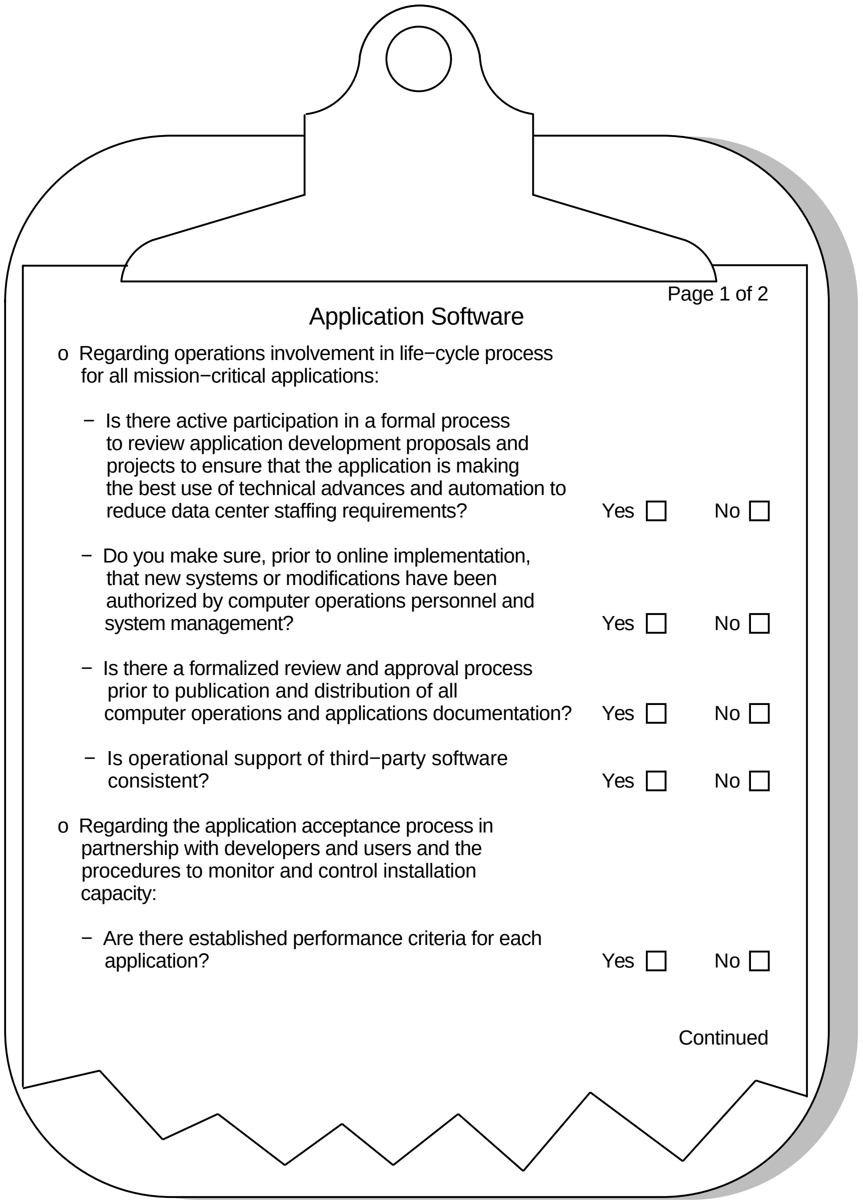
Page 8 of 8

Security (Cont.)

o Is the tool set planned or implemented?	Yes ☐	No ☐
o Is the percent of costs for backup and archiving less than 16% of total support costs?	Yes ☐	No ☐
o Is a program to reduce backup and archiving volumes and costs planned or implemented?	Yes ☐	No ☐
o Is the number of monthly failures fewer than 5 (failure means user access or activity is delayed for more than 30 minutes)?	Yes ☐	No ☐
o Is the mail menu response time less than 4 seconds (average, measured in seconds)?	Yes ☐	No ☐
o Are system capacity and utilization criteria and measurements planned or implemented?	Yes ☐	No ☐
o Is migration to the latest hardware and released software upgrades planned and implemented when justified?	Yes ☐	No ☐
o Is there involvement in product improvement through participation in field tests or other feedback to Digital engineering?	Yes ☐	No ☐
o Are data center policies and operating instructions regularly maintained and made available to employees responsible for planning or maintaining the computing environment?	Yes ☐	No ☐
o Is the coverage of the Digital service contracts sufficient to ensure minimum down time for critical systems and time periods (fiscal closings, development schedules, and so on)?	Yes ☐	No ☐

ZK−3849A−GE

A.5 Application Software Checklist



Page 1 of 2

Application Software

o Regarding operations involvement in life−cycle process for all mission−critical applications:

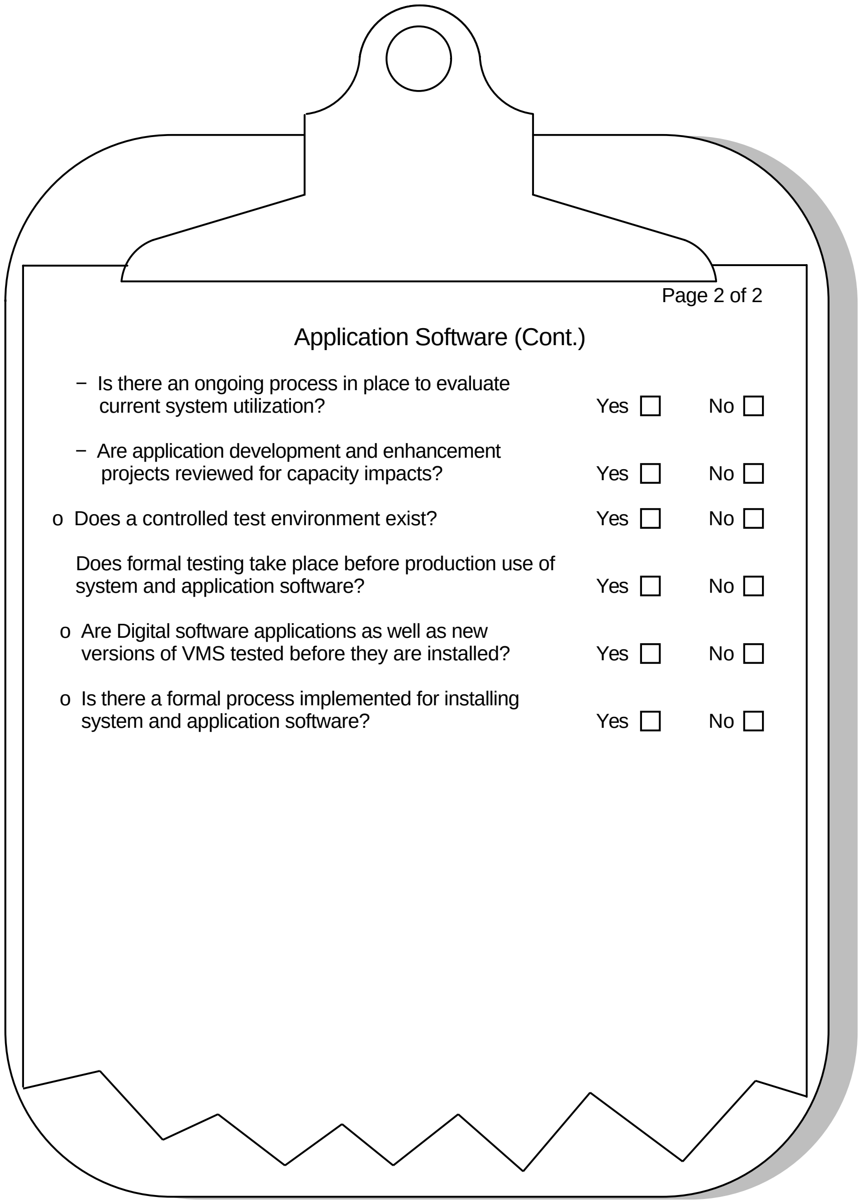
− Is there active participation in a formal process to review application development proposals and projects to ensure that the application is making the best use of technical advances and automation to reduce data center staffing requirements?	Yes ☐	No ☐
− Do you make sure, prior to online implementation, that new systems or modifications have been authorized by computer operations personnel and system management?	Yes ☐	No ☐
− Is there a formalized review and approval process prior to publication and distribution of all computer operations and applications documentation?	Yes ☐	No ☐
− Is operational support of third−party software consistent?	Yes ☐	No ☐

o Regarding the application acceptance process in partnership with developers and users and the procedures to monitor and control installation capacity:

− Are there established performance criteria for each application?	Yes ☐	No ☐

Continued

ZK−3850A−GE



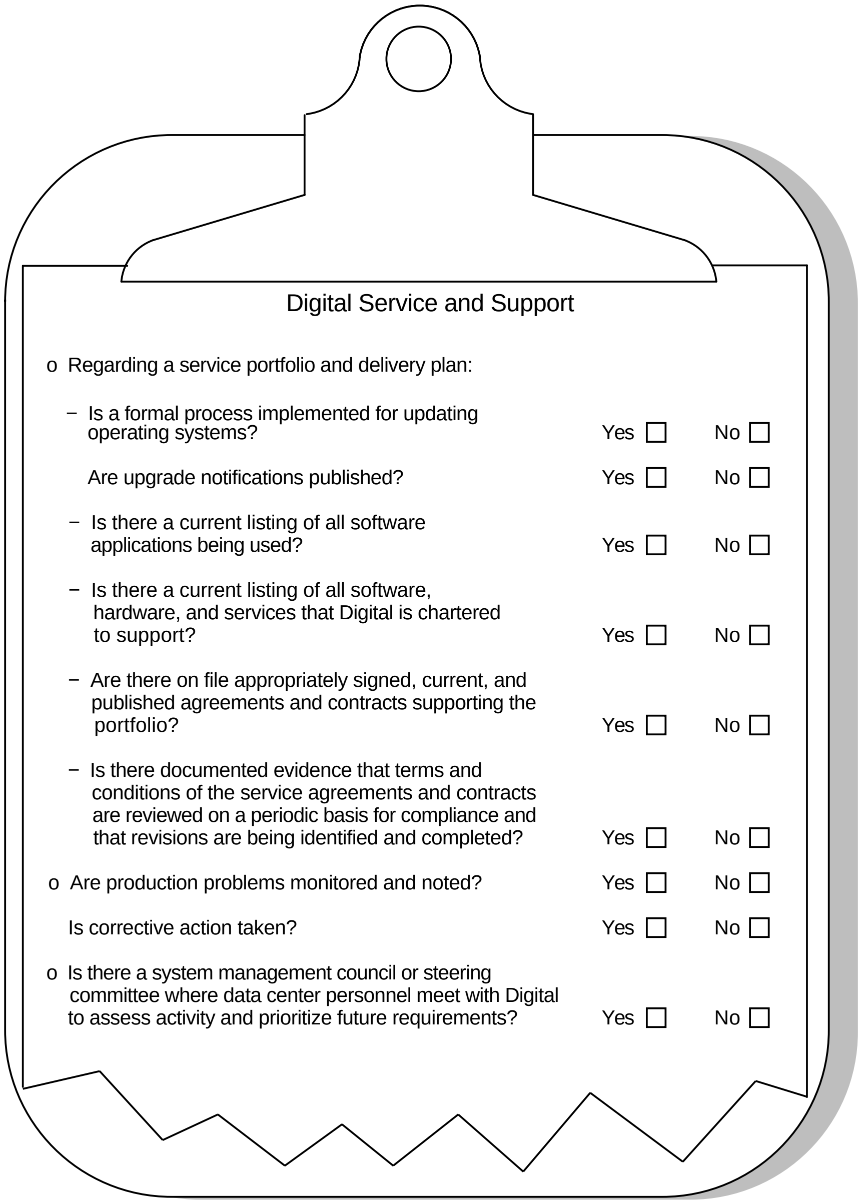
Page 2 of 2

Application Software (Cont.)

− Is there an ongoing process in place to evaluate current system utilization?	Yes ☐	No ☐
− Are application development and enhancement projects reviewed for capacity impacts?	Yes ☐	No ☐
o Does a controlled test environment exist?	Yes ☐	No ☐
Does formal testing take place before production use of system and application software?	Yes ☐	No ☐
o Are Digital software applications as well as new versions of VMS tested before they are installed?	Yes ☐	No ☐
o Is there a formal process implemented for installing system and application software?	Yes ☐	No ☐

ZK−3851A−GE

A.6 Digital Service and Support Checklist



Digital Service and Support

o Regarding a service portfolio and delivery plan:

− Is a formal process implemented for updating operating systems?	Yes ☐	No ☐
Are upgrade notifications published?	Yes ☐	No ☐
− Is there a current listing of all software applications being used?	Yes ☐	No ☐
− Is there a current listing of all software, hardware, and services that Digital is chartered to support?	Yes ☐	No ☐
− Are there on file appropriately signed, current, and published agreements and contracts supporting the portfolio?	Yes ☐	No ☐
− Is there documented evidence that terms and conditions of the service agreements and contracts are reviewed on a periodic basis for compliance and that revisions are being identified and completed?	Yes ☐	No ☐
o Are production problems monitored and noted?	Yes ☐	No ☐
Is corrective action taken?	Yes ☐	No ☐
o Is there a system management council or steering committee where data center personnel meet with Digital to assess activity and prioritize future requirements?	Yes ☐	No ☐

ZK−3852A−GE

A.7 Compliance Summary

Table A–1 provides a method for you to compile the results of the evaluation checklists. The table helps you to compare the number of times that you answered *Yes* to the total number possible. By examining areas where you are deficient, you can define what traits are lacking from your computing system and plan a strategy for achieving a desirable level of dependability.

Table A–1 Compliance Summary

Category	Total Possible Compliance	Yes Answers	% Compliant
General planning	15		
Environmental management	34		
Data center organization	12		
Security	78		
Application software	11		
Digital service and support	9		
Total compliance	159		

B Bibliography

This bibliography lists suggested readings for further study of topics related to system dependability. It is organized into two sections:

- Publications from Digital
- Publications from other sources

B.1 Digital Publications

The following Digital publications may be ordered by contacting DECdirect at 1-800-DIGITAL (1-800-344-4825):

Computer Engineering
A DEC View of Hardware Systems Design
C. Gordon Bell—J. Craig Mudge—John E. McNamara
Digital Press, Bedford MA
ISBN 0-932376-00-2
A course in understanding computer hardware through classic examples.

Computer Programming and Architecture
The VAX
Henry M. Levy—Richard H. Eckhouse Jr.
Digital Press, Bedford MA
A comprehensive study of VAX hardware architecture and OpenVMS operating system internals.

The Digital Guide to Software Development
Corporate User Publications Group
Digital Press, Bedford MA
An overview of Digital’s standard methodology.

The Human Factor
Designing Computer Systems for People
Richard Rubinstein—Harry M. Hersh—Henry Ledgard
Digital Press, Bedford MA
Helps you look at your system from the end-user's viewpoint.

Introduction to VAXcluster Application Design
Order Number: AA-JP32B-TE, with binder
Digital Equipment Corporation, Maynard, MA
For system designers and programmers who want to learn by example the techniques of building robust VAXcluster applications.

VAX/VMS Writing Real Programs in DCL
Paul C. Anagnostopoulos
Digital Press, Bedford MA
ISBN 1-55558-023-8
A style guide for implementing robust command procedures.

Version 5.4 VAXcluster Principles Update
Order Number: EK-VAXCP-UP-001
Digital Equipment Corporation, Maynard, MA
Describes how VAX VMS implements the distributed operating system called a VAXcluster.

Digital Technical Journal articles
Digital Equipment Corporation, Maynard MA
In-depth technical discussions by the responsible project engineers.

Availability in VAXcluster Systems and
Network Performance and Adapters, EY-H890E-DP

Compound Document Architecture, EY-C196E-DP

CVAX-based Systems, EY-6742E-DP

DECwindows™ Program, EY-E756E-DP

Distributed Systems, EY-C179E-DP

Fiber Distributed Data Interface, EY-H876E-DP

Software Productivity Tools, EY-8259E-DP

Storage Technology, EY-C166E-DP

Transaction Processing, Databases, and Fault-tolerant Systems, EY-F588E-DP

VAX 6000 Model 400 System, EY-C197E-DP

VAXcluster Systems, EY-8258E-DP

The following document is available but may be ordered only through your Digital representative:

The Digital Services Reference Guide
Order number: EC-G1726-76
Corporate Services Communications
Digital Equipment Corporation, Maynard, MA
A comprehensive handbook covering Digital's U.S. consulting services.

B.2 Other Publications

The following publications are available from sources other than Digital:

Out of the Crisis
W. Edwards Deming
MIT CAES, Cambridge
ISBN 0-911379-01-0
From one of the founding fathers of today's Quality Revolution.

Managing the Software Process
Watts S. Humphrey
Addison-Wesley, Reading, MA
ISBN 0-201-18095-2
The definitive text on improving your software development process.

Computer Disaster Recovery Planning
Kenneth N. Meyers
Battelle Memorial Institute
To help you prepare for the worst.

The Psychology of Computer Programming
Gerald M. Weinberg
Van Nostrand Reinhold Company, NYC
ISBN 0-442-29264-3
A classic treatise on the human side of software development.

Glossary

This glossary defines terms related to the dependability of computing systems.

A 24x365 computing environment

Computing performed 24 hours a day, 365 days a year.

availability

The percentage or amount of scheduled time that a computing system provides application service.

backup switching

An event that occurs when a second computing system or component picks up processing in the event that the primary computing system or component fails.

Business Recovery Server

A system integration product consisting of a license, system management tools, documentation, and installation and customization services that let you create a multiple-site data center VMScluster system.

client

Hardware or software that obtains a specific set of services from a server.

client/server

A style of computing that uses a client and a server. See also *client* and *server*.

computing component

Part of the total computing system around which an arbitrary boundary has been defined. The boundary can be defined at any level.

continuous improvement process

A methodology for implementing a process to continuously improve the dependability of your computing system.

database administrator (DBA)

A person responsible for the design, creation, maintenance, and tuning of a database.

data integrity

The ability of a system to maintain its information in a consistent state.

degraded performance

Performance in which service continues but response time is extended, the number of users that can be served is reduced, or both.

dependable computing system

A system that can be counted on to provide services to its users when those services are needed and with sufficient performance. The computing components are created and combined in the manner necessary to provide a required level of trustworthiness.

disaster tolerant computing

A computing style where the systems continue to provide production work regardless of fires, earthquakes, or other catastrophic disasters.

disk fragmentation

The state in which data file segments are so scattered across a disk that I/O access performance diminshes to the point where the disk might appear to be unavailable.

down time

The percentage or amount of time a computing system does not provide application service as scheduled.

error

An event during the operation of a computing component that produces incorrect results due to one or more faults. Errors are observed as incorrect responses within a specific computing component.

error correction

The action necessary to isolate the effects of faults to a specific computing component. The goal is to contain the impact of the problem.

failover

The ability of a computing system to reconfigure itself to use a working component when a similar component fails.

failure

The inability of a computing component to perform its function correctly due to one or more internal faults whose effects cannot be contained. Failures are observed by the consumers of the computing component's services as incorrect behavior.

failure recovery

The action necessary to restore the failed computing component to a correctly functioning condition. The goal is prompt return to zero defects.

fault

A defect in some component of a computing system.

fault management

The discipline used to engineer systems with a cost-effective balance of fault prevention qualities, error correction capabilities, and failure recovery facilities. Fault management is realized in the implementation of a dependable computing system. It is also a philosophy that is followed during the implementation.

fault prevention

The process of designing and constructing computing components to be free from faults. The goal is zero defects.

fault tolerance

The ability of a computing system to withstand faults and errors while continuing to provide the required services. See also *hardware-based fault tolerance* and *software-based fault tolerance.*

front-end

The part of a computing system that typically handles data capture, terminal displays, communications, and validation functions.

hardware-based fault tolerance

The ability to detect, isolate, and bypass a fault; engineered into and executed through hardware.

hot standby

A second running computing system that is ready to pick up application processing in the event that the primary computing system fails. That is, the secondary system takes over the processing at the point where the original computing system stopped and the secondary system continues the processing.

lobe

A set of CPUs that are connected together by one or more VMScluster interconnects. A lobe must have its own system disk(s) used by all CPUs in that lobe. A single CPU with a local system disk can be a lobe. Not all CPUs are part of a lobe, however. For example, a set of workstations is not a lobe.

lock step

A characteristic when two or more CPUs execute exactly the same instructions and all complete the instructions before any CPUs start their next instruction.

MDF

Multi-Datacenter Facility has been renamed Business Recovery Server. See the Business Recovery Server definition.

mean-time-between-failures (MTBF)

The average time that passes before a computing component fails such that remedial action is required.

MTBF

See mean-time-between-failures.

namespace

The collection of unique names in the VAX Distributed Name Service (DNS) database. Each name refers to a particular network resource.

reconfiguring

The ability of a computing system to use different sets of components and connections to form a new configuration.

recoverability

The ability of a system to reconfigure itself and continue to (or quickly resume) operation.

redundant component

Duplicate or extra computing components that protect a computing system from failure.

reliability

The ability of a computing system to operate without failing. Measured by mean-time-between-failures (MTBF).

scalability

A measure of how well the software or hardware product is able to adapt to future business needs.

server

Hardware or software that provides a specific set of services to a client.

single points of failure

Portions of a computing system that, if they fail, cause the system to cease providing service.

SMP

See *symmetric multiprocessing (SMP)*.

software-based fault tolerance

The ability to detect, isolate, and bypass faults; executed through software.

symmetric multiprocessing (SMP)

A multiprocessing system configuration in which all processors have equal access to operating system code residing in shared memory and can perform all, or almost all, system tasks.

throughput

The number of transactions or jobs a computing system can complete in a given period of time.

transition time

The amount of time that a failed computing system takes to reconfigure itself following a component failure. Experienced as a gap in user response time.

transparent failover

The ability of a computing system to reconfigure itself or to switch processing to a redundant component and to continue processing without writing any user code or taking any corrective action.

uninterrupted service

The ability of a computing system to continue providing application service during and after component failure without interruption or perceptible pause.

Y-connector

Hardware that joins two synchronous communications lines into a single output line.

zone

A section of a fully configured VAXft fault tolerant computing system that contains a minimum of a CPU module, memory module, I/O module, and associated devices. A VAXft system consists of two such zones with synchronized processor operations. If one zone fails, processing continues uninterrupted through automatic failover to the other zone.

Index