DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Managing AOS/VS and AOS/VS II

093-000541-03

Managing AOS/VS and AOS/VS II

093-000541-03

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000541

Printed in the United States of America
Rev. 03, June 1993

Notice

DATA GENERAL CORPORATION
4400 Computer Drive
Westboro, MA 01580

Managing AOS/VS and AOS/VS II
093-000541-03

Revision History:	Effective with:
Original Release — December 1988	
Addendum 086-000141-00 — June 1989	
First Revision — March 1990	
Second Revision — December 1991	AOS/VS II Revision 2.10 and AOS/VS Revision 7.69
Addendum 086-000193-00 — June 1992	AOS/VS II Rev. 2.20 and AOS/VS Rev. 7.70
Third Revision — June 1993	AOS/VS II Revision 3.00 and AOS/VS Revision 7.71

A vertical bar in the margin of a page indicates substantive technical change from the previous revision. The only exception is Chapter 6, which consists of entirely new material.

Preface

AOS/VS and AOS/VS II are two of Data General Corporation's (DGC) proprietary operating systems for 32–bit ECLIPSE® MV/Family and DS–series computer systems.

Both AOS/VS and AOS/VS II can be managed in two very different ways, as a regular system or as a preinstalled system. Regular systems provide a very high degree of flexibility, but assume a higher degree of technical knowledge. Preinstalled systems are easier to use. If you are running preinstalled AOS/VS, Model 31133, or preinstalled AOS/VS II, you should be reading either *Starting and Updating Preinstalled AOS/VS* (069–000293), *Starting and Updating Preinstalled AOS/VS II* (069–000294), *Starting and Updating Preinstalled AOS/VS on ECLIPSE® MV/3000 DC and ECLIPSE MV/5000™ DC Series Systems* (069–000481), or *Starting and Updating Preinstalled AOS/VS II on ECLIPSE® MV/3000 DC and ECLIPSE MV/5000™ DC Series Systems* (069–000480) as your primary manual. You can also choose to use the System Management Interface (SMI) as your interface to the system. The manuals describing the SMI are *Using the AOS/VS System Management Interface (SMI)* (069–000203) and *Using the AOS/VS II System Management Interface (SMI)* (069–000311).

In this manual we assume that you are running a regular AOS/VS or AOS/VS II system and that you have already generated and brought up your initial AOS/VS or AOS/VS II system, as described in the manuals *Installing, Starting, and Stopping AOS/VS* (093–000675) and *Installing, Starting, and Stopping AOS/VS II* (093–000539). Those manuals got you up and running by showing you how to install, generate, start, and stop your system; bring up a starter system; generate a tailored system; bring up the multiuser environment; and routinely start up and shut down your system.

As their names imply, AOS/VS and AOS/VS II have a lot in common, and this manual covers the common ground you as a system manager or operator will want to understand: creating user profiles, creating and terminating the EXEC process, backing up and restoring files, using operating system runtime tools to monitor and improve system performance, and establishing a level of security appropriate for your users and your site.

This manual is organized as a reference book. The chapters describe the tools available to help you manage your operating system. This manual also explains when and how to file a Software Trouble Report or STR.

Manual Organization

The manual is organized as follows:

Chapter 1	provides an overview of AOS/VS and AOS/VS II and what it means to manage either operating system.
Chapter 2	explains the user profile editor, PREDITOR. You will use this editor when you create, edit, or delete a user profile.
Chapter 3	details the EXEC multiuser management program.
Chapter 4	explains various backup utilities and issues involved in backing up user and system files.
Chapter 5	shows how to back up and restore AOS/VS and AOS/VS II files using the CLI commands DUMP and LOAD or the DUMP_II and LOAD_II utilities.
Chapter 6	shows how to use the FSCOPY utility to back up AOS/VS II LDUs and to restore LDUs or Files. FSCOPY is part of AOS/VS II Revision 3.01.
Chapter 7	shows how to use the LDCOPY utility to back up and restore AOS/VS II LDUs.
Chapter 8	shows how to use the MSCOPY utility to back up and restore AOS/VS files.
Chapter 9	shows how to use the PCOPY utility to back up and restore AOS/VS LDUs.
Chapter 10	gives some background about planning for and implementing high availability, including hardware and software mirroring.
Chapter 11	describes runtime tools, including the OP CLI, LOCK_CLI, PED, SYSLOG, CON0 logging, Superuser logging, REPORT, CONTEST, DISCO, LDUINFO, and SPRED.
Chapter 12	explains how to submit Software Trouble Reports (STRs).
Chapter 13	discusses performance issues: process types, changing PID–size type, multiple–processor computers, classes and logical processors, disk space as it affects performance, data caching, and using the utilities HISTO and HISTOREPORT and LOGCALLS.
Chapter 14	reviews system security. It explains how to safeguard user passwords, sensitive material, and hardware from unauthorized access.
Appendix A	details operating system modem support.
Appendix B	describes the format of MAPPER files.

What About Peripherals?

Peripherals include disk units, tape and/or diskette units, the system console, user terminals, and letter–quality, laser, and line printers.

Before you can begin, at least one disk unit, one tape or diskette unit, and a system console (DASHER® terminal) must be connected to your computer, and all must have adequate power.

Related Documentation

This section describes manuals that you might find helpful additions to this one.

Hardware Operation

To run an operating system, you must know how to use the switches and controls on your computer, tape and disk units, system console, and printer(s). See the appropriate documentation from the list that follows:

- *DASHER® Operator Reference Series* manual for your system console.
- 014–series booklets for your disk unit(s); or, for older units, *DGC Disc Drives* (014–000099).
- *Magnetic Tape Transports* (014–000095). This operator reference series book describes how to operate a tape unit.
- *DGC Line Printers* (014–000089). Describes how to operate line printers.

For device status errors and other information, you might want to refer to the manual of the appropriate device.

Hardware Error Diagnostics and System Control Program (SCP)

The Advanced Diagnostic EXecutive (ADEX) system is an optional diagnostics package for MV/Family systems. ADEX provides a complete suite of diagnostics, including tests for peripherals and system exercisers for Data General hardware. ADEX is shipped on tape or diskettes, and it can be installed on the system disk along with AOS/VS or AOS/VS II. It provides easy, fast access to hardware diagnostics from a system startup menu. For ADEX information, see the following manual:

- *ADEX Operator's Manual* (014–000744). Describes how to install and run ADEX.

Many MV/Family computers allow remote diagnostic testing. If your computer permits this, and you have a contract with DGC that supports it, see the following manual:

- *Communications Switch–II User Operation and Installation Guide* (015–000207).

The System Control Program (SCP) features that you need for routine operation are covered in this manual. For more information, refer to the SCP manual that came with your computer.

AOS/VS, AOS/VS II, and Related Software

This manual explains only a few tasks involved in maintaining and using the AOS/VS and AOS/VS II operating systems. Some other tasks, and the manuals that explain them, are

- *AOS/VS and AOS/VS II Error and Status Messages* (093–000540) describes all the operating system error messages, including those you might encounter during system installation, startup, and shutdown.
- *AOS/VS and AOS/VS II Glossary* (069–000231–01) explains important terms and concepts.
- *Installing, Starting, and Stopping AOS/VS* (093–000675) and *Installing, Starting, and Stopping AOS/VS II* (093–000539) explain first–time issues. They are companion manuals and normally precede the manual you are reading now.
- *Learning to Use Your AOS/VS System* (069–000031) and *SED Text Editor User's Manual* (093–000249) describe using the SED editor shipped with the operating system.
- *Managing and Operating the XODIAC™ Network Management System* (093–000260) describes how to install and manage the Data General proprietary network software.
- *Managing the CEO® System* (093–000286) describes how to install and manage the CEO office automation software.
- *Using the CLI (AOS/VS and AOS/VS II)* (093–000646) describes the AOS/VS and AOS/VS II file and directory structure and how to use the CLI, a command line interpreter, as the interface to the operating system.

Refer to the Document Set after the Index for a complete, annotated list of AOS/VS and AOS/VS II documentation.

You might also need the pertinent networking manuals if your system is connected to other computers in a network. If you have bought the AOS/VS Performance Package — which includes the Class Assignment and Scheduling Package (CLASP), a separate product — you will want to see the accompanying manuals.

After you get your system and multiuser environment up and running, you will want to run other software, like compilers and data management products. These are described in books shipped with the software.

Update and Release Notices

AOS/VS and AOS/VS II Release and Update Notices have the latest details on all system software, including enhancements and changes, notes and warnings. Notices are supplied both as printed listings and as disk files that you can print. The filename in directory :UTIL of the AOS/VS Model 3900 Update Notice is 078_000105_**; that of the AOS/VS II Release Notice is 085_000930_**; and that of the AOS/VS II Update Notice is 078_000374_**. Suffixes (**) change with each revision.

You should read the Update and Release Notices. If you want to know the features of a release, or have problems with a release, check the notice for solutions. The notices assume that you know the operating system well — so parts of them may be difficult to understand until you *do* know the system.

Where we have not provided new printed documentation, documentation–changes files, also in :UTIL, are part of each release or update, but you must print these yourself after installing the new software. The filenames have the form 0ss_nnnnnn_rr, where ss is the series, nnnnnn is the part number, and rr is the revision (for example, 093_000541_02 would be the documentation–changes file for this manual). We suggest that, as you receive new software revisions from DG, you print the documentation–changes file(s) and update the manual(s) as needed.

The Newsletter

Finally, you will find the *AOS/VS Monthly Newsletter* a useful source of information on the latest enhancements to both AOS/VS and AOS/VS II.

Reader, Please Note:

We use these conventions for command formats in this manual:

REQUIRED required *[optional]* ...

Where	Means
REQUIRED	You must type the uppercase word, such as a command (or its accepted abbreviation), as shown.
required	You must type an argument, filename, or other variable in place of the lowercase word or letter. For example, the x in @MTx0 can be B, C, D, or J, depending on the type of magnetic tape drive. Sometimes we show { required1 / required2 } which means you must type *one* of the arguments. Do not type the braces; they only set off the choice.
[optional]	You have the option of typing this argument. Do not type the brackets; they only set off what is optional.

... You may repeat the preceding entry or entries. The explanation will tell you exactly what you may repeat.

Additionally, we use certain symbols in special ways:

Symbol	Means
⤶	Press the NEW LINE key on your terminal's keyboard. If there is no NEW LINE key, press the Carriage Return (CR) key.
)	The AOS/VS and AOS/VS II operating system CLI prompt.
Su)	The AOS/VS and AOS/VS II CLI Superuser prompt.
SCP–CLI>	The AOS/VS and AOS/VS II SCP CLI prompt.

All numbers are decimal, except for device codes and numbers marked octal. For example

27 buffers	means 27 decimal
device code 27	means 27 octal
27 octal	means 27 octal

We show CLI and SCP CLI commands in UPPERCASE; but you can type them in lowercase, uppercase, or any combination. Finally, we use

This typeface to show your entry.
This typeface to show system queries and responses.
`This typeface to show listings and status displays.`

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the North American Data General Customer Support Center (CSC) by calling 1–800–DG–HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive FOCUS monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1–800–253–3902 or 1–508–443–3330.

End of Preface

Contents

Chapter 1 — Overview

Chapter 2 — Editing User Profiles with PREDITOR

Chapter 3 — Managing User Processes with EXEC

Chapter 4 — Choosing a Backup Strategy

Chapter 5 — Using DUMP/DUMP_II and LOAD/LOAD_II to Back Up and Restore Files

Chapter 6 — Using FSCOPY to Back Up AOS/VS II LDUs and to Restore LDUs or Files

Chapter 7 — Using LDCOPY to Back Up and Restore AOS/VS II LDUs

Chapter 8 — Using MSCOPY to Back Up and Restore AOS/VS Files

Chapter 9 — Using PCOPY to Back Up and Restore AOS/VS LDUs

Chapter 10 — Improving System Availability

Chapter 11 — Using Other Runtime Tools

Chapter 12 — Submitting a Software Trouble Report (STR)

Chapter 13 — Fine–Tuning System Performance

Chapter 14 — Maintaining System Security

Appendix A — Modem Support

Appendix B — XLPT Mapper Files

Tables

Table

Figures

Figure

Figure

Chapter 1
Overview

This chapter introduces you to the AOS/VS and AOS/VS II operating systems and to the procedures involved in managing them. Read it when

- You want a brief overview of AOS/VS or AOS/VS II; or
- You want to know about the areas of responsibility of an AOS/VS or AOS/VS II system manager

AOS/VS and AOS/VS II have many features in common but they have important differences as well. For example, you format disks and create logical disk units (LDUs) differently and follow different installation procedures. For these reasons, Data General has produced different manuals for installing, starting and stopping AOS/VS and AOS/VS II. But there are important similarities. Because many facets of system management are the same for the two operating systems, this manual covers both operating systems. In this manual, wherever a difference in features or behavior of the two operating systems exists, we will refer to AOS/VS and/or AOS/VS II by name. Where features and behavior of the two operating systems are the same, we will use the term *operating system*.

The major sections in this chapter are

- What is AOS/VS?
- What is AOS/VS II?
- Using the Command Line Interpreter (CLI)
- The Operating System File Structure
- What's Involved in System Management?
- Locating Information About Your Operating System

What Is AOS/VS?

AOS/VS is a multitasking, multiprogramming, demand–paged, virtual storage operating system. You can use it to support users on a time–sharing basis, to run batch jobs, or to perform control applications on a real–time basis. You communicate with AOS/VS through either the Command Line Interpreter (CLI) or the System Management Interface (SMI). For complete information about bringing up the AOS/VS operating system on the ECLIPSE ® MV/Family of computers, read the manual *Installing, Starting, and Stopping AOS/VS*. There are two different models of AOS/VS, and this manual assumes you are running Model 3900, regular AOS/VS. It does not describe Model 31133, preinstalled AOS/VS.

What Is AOS/VS II?

AOS/VS II is an enhanced version of the AOS/VS operating system. Like AOS/VS, AOS/VS II is a multitasking, multiprogramming, demand–paged, virtual storage operating system for Data General's ECLIPSE MV/Family and DS–series of 32–bit computers.

AOS/VS II offers increased reliability and data availability by providing a file system structure that is compatible with earlier revisions of the AOS/VS operating system but that differs from AOS/VS from a system manager's point of view. Under AOS/VS II, you use a single, menu–driven utility, Disk Jockey, to manage all the disk–related functions of the operating system.

AOS/VS II also provides enhanced support for several optional communications products. AOS/VS II TCP/IP, AOS/VS II ONC™/NFS® Services, DG/OTS, and XTS II provide libraries of functions that a system manager links into AOS/VS II when running VSGEN.

For more information about bringing up the AOS/VS II operating system on the MV/Family of computers, read the manual *Installing, Starting, and Stopping AOS/VS II*.

Using the Command Line Interpreter (CLI)

You can communicate with AOS/VS II by typing Command Line Interpreter (CLI) commands on a terminal, or via the System Management Interface (SMI), but we assume that you are using the CLI as the interface to the operating system. The CLI is an interface programming language with built–in commands and pseudomacros. Pseudomacros are system commands that either return a value or create additional expressions.

For most system configurations the CLI will begin running as soon as you log on. The CLI commands give you access to most elements of your system. You can control peripheral devices, such as line printers and tape drives, using CLI commands. You can also use the CLI to create, delete, and move files, as well as execute programs and utilities.

If you are not already familiar with the CLI, refer to the manual *Using The CLI (AOS/VS and AOS/VS II)* for more information on how to use it.

(You may want to explore using the SMI if you will manage your system infrequently. For complete information about using the SMI with your operating system, read either *Using the AOS/VS System Management Interface (SMI)* or *Using the AOS/VS II System Management Interface (SMI)*.)

The Operating System File Structure

From a user's point of view, the AOS/VS and AOS/VS II file structures are practically the same. A typical file system under either operating system looks something like that shown in Figure 1–1.

The root directory (:) and other system directories are created and managed by the operating system or its utility programs. The operating system program file is in the directory :SYSGEN.

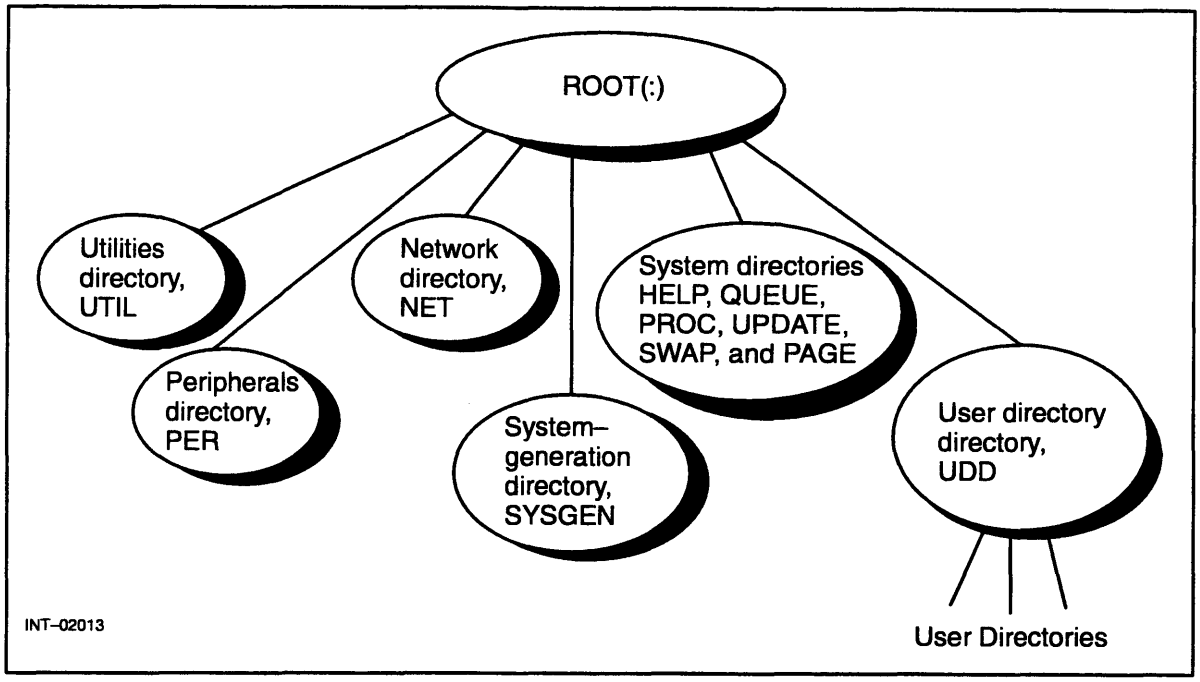


Figure 1–1 Operating System File Structure

The directory :UDD (stands for User Directory Directory) contains a subordinate directory for each time–sharing user. A user is an authorized person who can execute application programs. So, :UDD typically has many subordinate directories and uses a lot of storage space.

In addition to user directories, the file system contains system and application programs. While you can organize user directories in many ways, both operating systems and many system programs expect to find the system files in certain directories. Do not arbitrarily move system files into different directories or create links to system files. If you do, the operating system may not work as expected or may not work at all.

To keep your system simple and easy to use, you should make each physical disk a single–disk *logical disk unit* (LDU). In both AOS/VS and AOS/VS II, the system disk LDU must reside on a single disk. Under AOS/VS, LDUs occupy one or more entire physical disks. With AOS/VS II, an LDU can occupy *parts* of one or more physical disks; an LDU doesn't have to take up the entire disk. You might set up a multiple–disk LDU if you need to handle a very large file — perhaps a database file — that won't fit on a single disk. The system will then access the multiple–disk LDU as one directory file, providing enough space for the large database file. But, unless

you need to create complicated multipiece LDUs, you are better off making each physical disk a separate LDU.

First, plan the LDU structure of your physical disk: whether to have everything in one LDU, or to create separate LDUs on the same physical disk for your user directories (:UDD), swapping/paging directories (:SWAP and :PAGE), CEO files and so forth. How you set up LDUs on your operating system depends on the particular needs of your user community (which files are used most frequently) and the number and type of disk units available for storage. Once you know how you want your disks structured, under AOS/VS use the Disk Formatter and Installer utilities to format your physical disks and create the appropriate LDUs. These utilities are described in the manual *Installing, Starting, and Stopping AOS/VS*. Similarly, if you are running AOS/VS II, you'll use the Disk Jockey utility, described in the manual *Installing, Starting, and Stopping AOS/VS II*.

For an example of a single–disk LDU system, imagine the structure in Figure 1–1 on a single disk. For an example of a larger multiple–disk LDU system, see Figure 1–2.

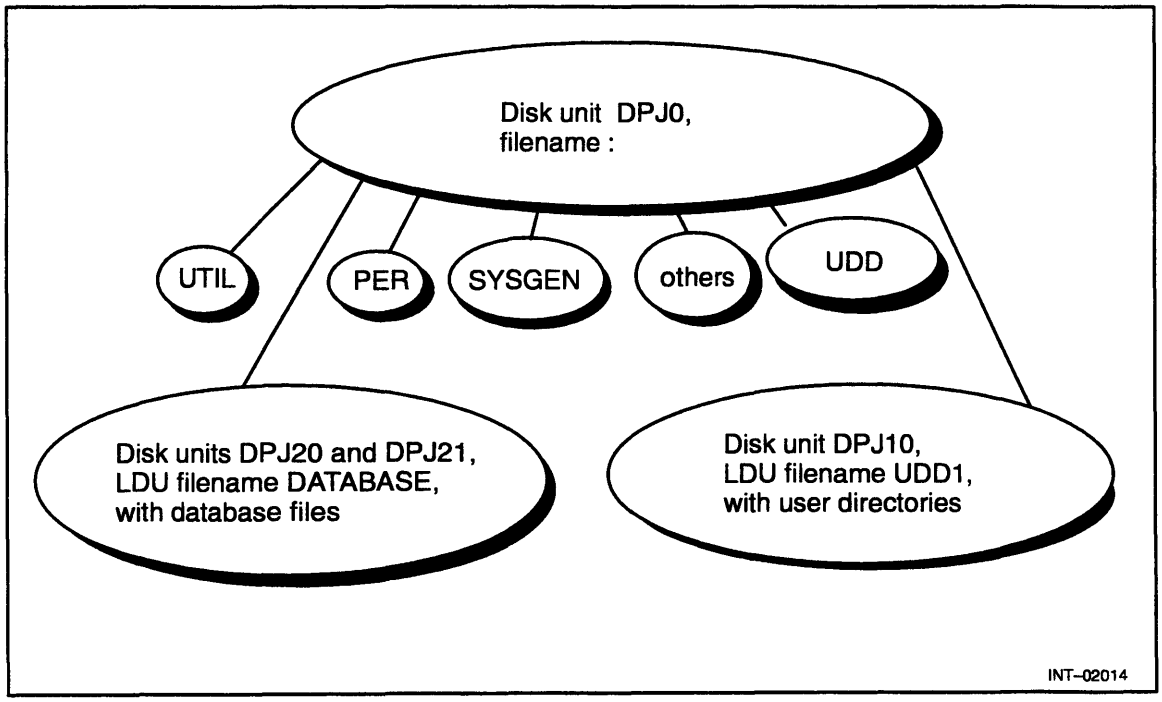


Figure 1–2 Multiple–Disk LDU System

First you will create a single–disk LDU in disk unit 0 on the first disk controller. This will be the system disk. But after you have formatted this LDU, you may need to format other physical disks (if you have other units) and make decisions about their names. Complete details of the system generation process for the regular model of AOS/VS or AOS/VS II can be found in the *Installing* manual for your particular operating system. Details of the system generation process for preinstalled models of AOS/VS and AOS/VS II can be found in the manuals *Starting and Updating Preinstalled AOS/VS* and *Starting and Updating Preinstalled AOS/VS II*.

What's Involved in System Management?

After you have installed, started, and tailored your particular operating system (as described in the *Installing* manual for your operating system), you'll need to determine how you are going to manage it.

You will need to make decisions about how the system can best serve your organization by making users productive. Managing a system is not as ordered a procedure as installing and starting the system. Each organization has its own definition of users, jobs, roles, development, and production. Many of these decisions involve other software, like data management systems, but there are a number of operating system options as well. Generally, managing the system includes tasks like

- Formatting physical disks, and creating, changing, and deleting logical disk units (LDUs);
- Helping the system's users: creating and changing profiles for users; starting up the EXEC process and ensuring that batch and device requests run smoothly, and perhaps doing preventive maintenance;
- Executing the installation's main application programs and bringing up network/communications software (if any) on schedule;
- Overseeing backup strategy and procedures;
- Monitoring and improving system performance;
- Determining how to run the system efficiently: when to monitor user processes and what steps to take to ensure system security;
- Knowing how to update your system and when to get in touch with Data General.

Formatting Physical Disks and Creating Logical Disk Units

An LDU is one or more physical disks (or parts of one or more physical disks) logically linked together. LDUs span one or more physical disks to provide storage for very large files or databases that cannot fit on a single physical disk.

If you are running AOS/VS, you will use the Disk Formatter utility to format one or more physical disks into one or more logical disk units (LDUs). The Disk Formatter utility allows you to do either a *Full* or a *Partial* format on a physical disk. A Full format ignores the AOS/VS file structure and writes a new bitmap on the LDU, effectively destroying all AOS/VS files on the disk. A Partial format retains the existing bitmap and uses read–only surface analysis when checking the disk surface for flaws. A Partial format does not destroy any AOS/VS files on the disk. The Disk Formatter utility is documented in the manual *Installing, Starting, and Stopping AOS/VS.*

If you are running AOS/VS II, you will use the Disk Jockey disk utility to perform all disk–related functions such as formatting physical disks and creating, changing, and deleting logical disk units (LDUs). Disk Jockey is described in the manual *Installing, Starting, and Stopping AOS/VS II.*

If you are migrating from the AOS/VS operating system to AOS/VS II, see the AOS/VS II Release Notice, which tells you how to load and print a text file, whose pathname is :UTIL:NEWFS_MIGRATION.DOC. Because of the differences in the AOS/VS and AOS/VS II file systems, you will need to dump all your AOS/VS files to tape, reformat your disks using Disk Jockey, and then reload your files. The procedure is basically simple but you must do it carefully. The file NEWFS_MIGRATION.DOC will help you plan and implement your migration. You will also need to see the manual *Installing, Starting, and Stopping AOS/VS II* to perform all disk–related functions such as formatting physical disks and creating, changing, and deleting logical disk units (LDUs).

Helping Users

A user is any person who can log on to the system, not only under EXEC, but under another Data General product like CEO, DG/SNA, or TCP/IP. A user can be an applications or systems programmer, a system operator, a manager, or a nontechnical person such as a data entry operator, a word processing typist, an executive seeking summary information, or a personnel file clerk.

Usually, organizations acquire computer systems to increase productivity. For your organization to realize this increase in productivity, users need to work effectively.

Anything you can do to make the system easier to use will increase the productivity of users. Many Data General products have Help function keys; others (like the CLI) have Help commands. You can provide users with easily accessible, pertinent Help messages and meaningful error messages for your own applications.

If response time seems slow at user terminals, think about running fewer batch streams. Use PED (discussed in Chapter 11 of this manual) to see how each process is behaving, and change the working set and I/O usage parameters of some processes, if necessary. ECLIPSE MV/Family systems are fast; you can take advantage of this by basing process parameters on users' primary requirements at any given time.

To get a CLI user's point of view, sit down at a user terminal with the manual *Learning to Use Your AOS/VS System*. It leads the reader through sessions with the CLI, text editors, FORTRAN 77, COBOL, Interactive COBOL, BASIC, Business BASIC, C, Pascal, and assembly language programming.

Executing Applications and Network Software

Every organization has major applications programs such as the CLI, SMI, or CEO that run almost continuously, and other applications software that needs to be run only occasionally, such as Data General's Class Assignment and Scheduling Package (CLASP). You will need to see to it that the organization–wide applications software for your site is run on schedule. You may also be the person to supervise the network software for your installation or your computer system.

A network enables two or more systems to communicate. It allows the computers to combine their assets and resources and to distribute their processing power among many users. A network is made of a local host, one or more links, and one or more remote hosts.

A local host sends information or a request. It can also receive information from the remote hosts.

A communications link connects the local host to the remote host. A link can be a wire, a telephone circuit, or any communications medium that transmits a signal. Remote hosts receive information or requests. Local and remote hosts can be in the same room or in different countries.

Performing Backups

One of the most important aspects of your job will be making sure user files get backed up regularly. If a disk fails or if a user accidentally deletes an important file, you will be called on to restore all files or the accidentally deleted file. We strongly recommend performing an incremental backup daily and a full backup weekly. An incremental backup involves dumping only those files which have been altered since the last backup was performed; a full backup dumps all files, certainly all user files but perhaps user and system files, regardless of whether they have been accessed or changed recently.

For most circumstances, we recommend that you use the DUMP_II utility, shipped with the operating system, to back up entire files. Chapter 4 explains how to use this utility and shows you the CLI macros for backup and restoration. For enhanced ease of restoration, you may want to use the DUMP_3 utility, a separate product with its own manual.

AOS/VS also offers you the ability to back up only the modified disk sectors of files. The utility MSCOPY notes only those files that have been modified since the last backup; this can be a great time–saver if your system has large database files. Chapter 5 describes the MSCOPY utility. Or, AOS/VS provides the PCOPY utility to back up entire LDUs. PCOPY does a physical copy of the LDU. Chapter 6 describes the PCOPY utility.

If you are running AOS/VS II, you can use the LDCOPY utility, which is part of the Disk Jockey disk management utility. You access LDCOPY from the Disk Jockey Main Menu. Chapter 7 explains the LDCOPY utility in detail.

Monitoring and Improving System Performance

An important part of managing a system is understanding how the operating system manages processes. For the most part, both operating systems manage their resources quite efficiently, giving memory and CPU time to interactive processes according to their types, priorities, and groups. Chapter 9 discusses operating system processes in detail and explains how you can improve system performance according to your own needs.

AOS/VS and AOS/VS II offer many tools, such as log files, for monitoring system performance and usage. The system log (SYSLOG) and the error log (ERROR_LOG) record user account and error information.

If you want to ensure accurate system logging, consider including the command to start SYSLOG in the UP.CLI macro that you create for your system. (See the *Installing* manual for your particular operating system for a detailed discussion of the UP.CLI macro.)

The system log file (SYSLOG) grows fast, so dump and delete it regularly. This is critically important with full detail logging. If the system runs out of log space, it may halt with a fatal error message. For logging to be worthwhile, reports should be generated and examined routinely for errors and signs of security violations. Chapter 11 describes logging.

Periodically, you may want to run CONTEST, the AOS/VS and AOS/VS II hardware test package, to identify potential problems. Chapter 9 explains the details about CONTEST. Preventive maintenance can help safeguard your hardware, and keep your whole system running smoothly.

Updating Your Operating System

AOS/VS and AOS/VS II revisions and updates include new functionality and fixes to known problems. Part of your job will be updating your operating system. Reading update and release notices with an eye to keeping your users happy will make your job better, too.

The manuals *Installing, Starting, and Stopping AOS/VS* and *Installing, Starting, and Stopping AOS/VS II* explain the details of how to install a release or update of your particular operating system.

Getting Help from Data General

Generally, when you buy a Data General system, the price includes a certain period of Field Engineering support, Software Subscription Service updates, and perhaps some training courses. After the initial period has expired, you must renew your service contract to ensure continued support.

In the United States, support people work from the Atlanta Service Center, and depending on your contract, you can call this center for help.

Outside the United States, the support people are often Data General system engineers; your sales representative can give you more information on support.

Where to Find Information About Your Operating System

Table 1–1 lists in alphabetical order some of the major tasks that concern a system manager, and the primary manual that has information on those tasks.

Table 1–1 Where to Find Information About Your Operating System

Topic	Where It's Described
Abnormal shutdown	The *Installing* manual for your operating system.
Backing up files	Chapters 4, 5, 6, 7, and 8 of this manual.
Batch	Chapters 3 and 4 of this manual
Bootstrapping	The *Installing* manual for your operating system.
Bringing up your first system	The *Installing* manual for your operating system.
CLI commands	*Using The CLI (AOS/VS and AOS/VS II)*; also Chapters 3, 4, 5, 11 and 14 of this manual.
Disks, formatting	The *Installing* manual for your operating system.
Disk Jockey utility	*Installing, Starting, and Stopping AOS/VS II*; also Chapters 7 and 10 of this manual.
Error conditions and messages	*AOS/VS and AOS/VS II Error and Status Messages*.
EXEC program	Chapter 3; also the *Installing* manual for your operating system.
Fixing LDUs	The *Installing* manual for your operating system.
Formatting LDUs	The *Installing* manual for your operating system.
Generating a system	The *Installing* manual for your operating system.
High availability features	Chapter 10 of this manual.
Installing microcode or a starter system	The *Installing* manual for your operating system.
Labeled tapes and diskettes	Chapters 4, 5, 8 and 9 of this manual.
Log file, system	Chapters 11, 13 and 14 of this manual.
Management decisions	Chapters 2, 3, 11, 13, and 14 of this manual.
Mirroring, hardware/software	Chapter 10 of this manual.

(continued)

Table 1–1 Where to Find Information About Your Operating System

Topic	Where It's Described
Mounting tapes for users	Chapters 3, 4, and 5 of this manual.
Multiuser environment	Chapter 3 of this manual.
Polishing (fixing) LDUs	*Installing, Starting, and Stopping AOS/VS II.*
Queues (EXEC)	Chapter 3 of this manual.
Runtime tools	Chapter 11 of this manual.
SCP operating system	The *Installing* manual for your operating system; also Chapters 11 and 13 of this manual. Also see the hardware manual for your particular ECLIPSE MV/Family computer.
Security	Chapter 14 of this manual.
Shutdown, normal and abnormal	The *Installing* manual for your operating system.
Software Trouble Report	Chapter 12 of this manual.
System performance	Chapter 13 of this manual.
System startup	The *Installing* manual for your operating system.
Updating the operating system	The *Installing* manual for your operating system.
User profiles (PREDITOR)	Chapter 2 of this manual.

(concluded)

End of Chapter

Chapter 2
Editing User Profiles with PREDITOR

This chapter describes PREDITOR, the user profile editor supplied with AOS/VS and AOS/VS II. PREDITOR creates the user profiles that identify system users to EXEC, the operating system's multiuser environment, and allows them to log on and off.

Read this chapter

- When you want specific information on establishing privileges for users on your system;
- When you want to understand the privileges that you can give or withhold from a user.

PREDITOR profiles, along with EXEC, provide the base for the multiuser environment. Profiles allow only authorized persons to use the system, without trespassing on other persons' or system files.

If you created your system's multiuser environment (described in the *Installing* manual for your particular operating system), you already have some experience with PREDITOR. That manual describes how to create a privileged operator profile and many general–purpose user profiles. As a base for the user profiles, the manual had you edit PREDITOR's default profile; then use the new default values as a basis for each user profile.

This chapter describes all the PREDITOR commands. These commands allow you to create, edit, delete, and rename profiles, among other things. For ease of use, this chapter duplicates some of the material found in the *Installing* manual for your operating system. The major sections are

- About PREDITOR
- BYE command
- CREATE and EDIT commands, with comments on disk space control
- DELETE command
- HELP command
- LIST command
- QUESTION command
- RENAME command
- USE command

About PREDITOR

The PREDITOR program file, PREDITOR.PR, is in directory :UTIL. It is a privileged program: only users with Superuser privilege can run it, but they do not need to turn Superuser on in order to execute PREDITOR.

When it starts up, PREDITOR checks to see if the user directory directory (:UDD) exists. If :UDD does not exist, PREDITOR creates it. Then for every user profile you create, PREDITOR creates a user directory in :UDD, and the system creates a profile in the user profile directory, :UPD. The user directory and profile have the username you gave to PREDITOR. When you delete, edit, or rename a profile, PREDITOR makes the appropriate changes in file :UDD and has the system change the profile in :UPD.

When a person tries to log on a terminal or submits a batch job, EXEC checks the profile before it creates a user process to serve that person. If there is no valid profile, EXEC rejects the user or job. If there is a valid profile, EXEC creates a user process with the privileges defined in the profile. While the user process runs, it can use only these privileges. Thus, profiles are central to multiuser system operation.

User Access Control Lists (ACLs)

When you create a new profile or rename an old one, PREDITOR sets the access control list (ACL) of the user directory to username,OWARE. This ACL gives the user all access privileges to the directory. No one else, except a superuser, can access it at all. This ACL gives the user privacy, security, and unique access to all files and directories that he or she may create.

Executing PREDITOR

PREDITOR is in the directory :UTIL, so unless the working directory is :UTIL, your search list must include :UTIL. (The UP.CLI macro sets the working directory to :UTIL for the master CLI and its sons, so you can execute PREDITOR once the multiuser environment is up.) To execute PREDITOR, type

) XEQ PREDITOR ↵

PREDITOR will respond by display its banner message, and asking you what command you want to execute.

AOS/VS II User Profile Editor Rev n date time

Command:

PREDITOR is ready for a command.

If you want to change an answer that you previously gave to PREDITOR, press the uparrow key or press ^ (caret, Shift–6) until you reach the incorrect entry. Then type the desired answer and proceed. For example,

Use IPC [No] (Y, N, or NL) Y ↵	*Type your answer; then press NEW LINE.*
Password: ^	*Press the uparrow key to correct the previous entry.*
Use IPC [Yes] (Y, N, or NL) N ↵	*PREDITOR backs up, allowing you to change your answer.*

If, at any point, you want to return to the first PREDITOR question, press CTRL–C CTRL–A.

To exit from PREDITOR, type BYE and press NEW LINE.

Usernames, Passwords, and Network Access

The username is the identifier for every person who will use your system. The username is the only trace to the person who's responsible for the account. Usernames persist over long periods of time; they are not often changed (although PREDITOR does have a command to rename a profile).

Generally, your system should have a unique username for every person; more than one user should not use a single account. If, for any reason, you want to place a set of users in a group, think up a special identifier (like a suffix) for the usernames and make the identifier part of each username. Or, under AOS/VS II only with the 32–bit CLI, create groups of users in directory :GROUPS. Username groups are further explained in Chapter 14 in this manual.

The rest of this section explains how usernames and passwords relate to access over networks.

Data General network software you can run under AOS/VS and AOS/VS II includes TCP/IP (based on Ethernet/IEEE 802.3 protocol) and XODIAC™/XTS (based on X.25 protocol). With both networks, a user must know a valid username/password pair on a host system to access the other host system. ("Host" means the same thing as node or member.) Eventually you'll need to coordinate usernames and passwords with other host systems—to have profiles created for your users on these systems and/or learn which remote users to create profiles for.

For access to the network by means of TCP/IP, your choices of username and password have no functional effect. There's no functional benefit to having the same username and password on all hosts (although most people find it easier to remember a single username/password pair than several different ones).

For access to the network by means of XODIAC/XTS, your choices of username and password are significant. There's some functional benefit to having each person's username and password be the same on each system.

XODIAC/XTS software includes agents (applications) that provide different services. Using XTS to access network hardware, the agents are the Resource Management Agent (RMA), the Virtual Terminal Agent (VTA), and the File Transfer Agent (FTA). Details on these agents follow.

RMA The Resource Management Agent allows users on one host to access devices and files on another host. RMA allows network pathnames and remote access without a logon requirement; for example, you could issue the following command:

```
TYPE :NET:REMOTE_SYSTEM:UDD:SANDY:MYFILE ↵
```

For access to occur by means of RMA,

- The user must have a valid profile on both systems, *with the same user name and the same password* on both systems.
- The user must have the privilege *Access local devices from remote machines* on the remote host.
- The user must have access to the file or device he or she is trying to use. By default, each user has Owner access to files in his/her own user directory. Access to devices like tape units is governed by the unit's access control list (ACL) in the peripherals directory. Access to printers is generally governed by EXEC; users access print *queues* (not printers). You can control user access to queues and devices with the EXEC command ACCESS.

VTA The Virtual Terminal Agent lets users call remote hosts, log on, and use the remote host as if it were a local host. VTA is the only agent involved in PC networks, where a Data General system supports personal computers as if they were user terminals.

FTA The File Transfer Agent can transfer files from one host to another.

Agents VTA and FTA both require users to pass a logon procedure. For access to occur by means of VTA and FTA,

- The user must have a valid profile (but not necessarily the same username and password) on both systems.
- The user must have the privilege *Use virtual console* (for VTA) or *Access local resources from remote machines* (for FTA) on the remote system.
- The user must have appropriate access to the file or device. See comments under RMA.

For any agent, if the first two conditions aren't true, the user will receive an *Invalid username–password pair* message on attempted access.

NOTE: If you will use Data General's CEO® system over a network, remote printing no longer requires that the operator (OP) profiles match, effective with CEO Revision 3.00.

PREDITOR Username Templates

After profiles have been created, you can edit, delete, or list them by template. Username templates work much the same way as filename templates: + indicates all characters, – indicates all characters except a period, and * indicates one character. (But you can't use more than one template character at a time.)

PREDITOR does not sort usernames alphabetically. If you're editing and PREDITOR presents a profile you don't want to edit, you can press ^ (caret, Shift–6) at the *Password* question and PREDITOR will skip to the next username. For example,

) X PREDITOR ↵

...

Command: Edit ↵	Specify Edit.
Username: A+↵	Edit profiles that start with the letter A.
Username: ALLEN	PREDITOR displays profile name to edit.
Password change [No]? (Y or NL) ^	You don't want to edit Allen's profile; skip to the next user profile by pressing ^ (caret, Shift–6).
Username: ABBY	PREDITOR displays another profile name.
Password change? [No] (Y or NL) ↵	You want to edit this profile; press NEW LINE to continue.
...	Continue editing user profiles.

Used with the LIST command, a template tells PREDITOR to list all values in all matching profiles. With the DELETE command, PREDITOR displays each profile name for confirmation before deleting the profile.

You can cancel any template operation and return to the *Command:* prompt by typing CTRL–C CTRL–A.

BYE Command: Exiting from PREDITOR

Typing BYE and pressing NEW LINE terminates PREDITOR and returns you to the CLI. You can use BYE only when PREDITOR is asking for a command. (As with any program you can exit by aborting with CTRL–C CTRL–B, but a better solution is to use CTRL–C CTRL–A to return to the *Command:* prompt.) To exit from PREDITOR, type BYE at the prompt:

Command: BYE ↵

Terminating date time
)

CREATE and EDIT Commands: Creating or Editing a Profile

The CREATE and EDIT commands allow you to create a new profile and edit an existing one. After either command, PREDITOR asks a series of parameter questions. You can suppress questions temporarily with the QUESTION command.

PREDITOR displays default values in square brackets and valid answers in parenthesis after each question. For the CREATE command, the default answer to any question is the value in PREDITOR's !DEFAULT! profile, unless you edited the !DEFAULT! profile, or told PREDITOR to use another profile with a previous USE command. For the EDIT command, the default value is the current value in the profile you're editing. For example, the default value for the *Initial IPC file* question might say

Initial IPC file [:UTIL:LOGON.CLI] Change? (Y or NL)

which means that the default IPC file for this profile is :UTIL:LOGON.CLI and that you can answer Y to change it or press NEW LINE to accept the default.

When you edit a profile and change a value, the new value becomes effective the next time the user logs on. In other words, if you change a value while a user is logged on the system, the new value will not take effect until the user logs off and logs on again.

CREATE/EDIT Questions

When PREDITOR asks for a command, you can answer CREATE or EDIT. (The valid abbreviations are CR and E, respectively.) Then PREDITOR asks the following series of questions.

Username

When creating a profile, you must type a username; there is no default. Valid usernames are 1 through 15 valid filename characters long. When editing a profile, you can type a username or a template that includes one template character. PREDITOR templates are described in an earlier section. They work with the EDIT, DELETE, and LIST commands.

If users on another Data General system will use your system (or vice versa) over a network (for example, if your local CEO Electronic Mail and Calendar databases will be shared with remote DESKTOP GENERATION® system users), their username and password must be the same on all systems. For the CEO system, the CEO username and the operating system username must be the same on all systems. If the operating system username and CEO username (if any) aren't the same, and the password isn't the same, certain operations (like CEO mail or moving files) won't work. For a user who will simply log on another system in the network, the username and password can differ between systems.

If a network user's profile is renamed, it must also be renamed on other systems; the CEO profile (if relevant) must be recreated with a new username. Thus, in a network, renaming a profile might entail a lot of work. If a user changes a password locally, perhaps you can have him/her log on remote system(s) and change it there also.

PREDITOR contains a default profile under the username of !DEFAULT!. You can edit !DEFAULT! just as you can any profile to change default values. Editing !DEFAULT! can speed up the creation of real user profiles by allowing you to answer most questions by pressing NEW LINE. The new default values you place in !DEFAULT! remain only until PREDITOR terminates; then the original defaults return. !DEFAULT! is simply a convenience; no user profile or directory is associated with it.

When you want to edit a profile and can't remember the username, back up to the *Command:* prompt with ^ (caret, Shift–6). Then type BYE and press NEW LINE to exit, and display the names of all users by typing

) SUPERUSER ON ↵

Su) DIR :UDD ↵

Su) F/S ↵

... (CLI displays sorted names of all user directories) ...

Su)

NOTE: You have a choice of running either a 16–bit CLI or a 32–bit CLI. The two CLIs are compared as part of the discussion of the *Program [default] change?* question later on in this chapter.

Password change?

When you create a profile, you must give a password. When you edit a profile, you can type Y and press NEW LINE, and enter the new password or press NEW LINE to retain the old one. (If you do retain the old one, PREDITOR skips the next question.)

Each user must know his or her password to log on. So if you change it, be sure to let the user know. If a user forgets his/her password, you can simply enter a new one; you need not know the old password to change it with PREDITOR. A password must be 6 through 15 characters long and can be any combination of the following characters: upper– or lowercase letters (treated as uppercase), numbers 0 through 9, and all printing characters except uparrow.

By default, a user can change his or her password at logon by pressing the ERASE PAGE key or CTRL–L instead of NEW LINE after typing the old password.

Encrypt password [default]?

Password encryption is the process of converting a password into what appears to be a string of random characters. The system recognizes the random characters as the password, but a human reading the random characters cannot determine the password from them. PREDITOR can encrypt a password before storing it. From a security standpoint, this is desirable because no one — not even a superuser — can find out an encrypted password.

Once encrypted, a password can't be unencrypted. When the user changes passwords, the new password will also be encrypted. If a password is encrypted and you later decide to have it stored unencrypted, you must edit the profile, create a new password and type it, and say No to the *Encrypt password?* prompt. Then tell the user the new password. (He or she can change it if desired.)

It is a good idea to encrypt passwords. Encryption will *not* affect access by means of CEO Mail to remote hosts or CEO printing on remote printers, nor will it affect access by means of TCP/IP or XODIAC/XTS network software. So press NEW LINE to take the default, Y.

Initial IPC file [default] change?

The InterProcess Communication (IPC) is a file the system will execute when this user logs on. It exists to communicate with the user's initial program, to set the search list, default access control list, and so on. It is not required, but it is very useful.

Generally, the most flexible way to handle interprocess communication is to have one central IPC file that executes individual logon macros in each user's directory. The logon macro in each user's directory can have a specific sequence of CLI commands, which may execute another program like the CEO system or BASIC.

The central IPC filename can be the same for all users. Only the first 512 characters are executed, so the central file shouldn't exceed 512 characters. The user logon macro resides in each user's directory, pathname :UDD:username:filename. CLI users can create or edit the user logon macro file if they don't like the default values imposed by the central macro. The *Installing* manual for your particular operating system shows how to create a central logon macro and user macro, and how to put the user macro in user directories.

If you want a user to start in the CEO system, you can use the CEO–supplied macro CEO.STARTUP.CLI in the user's logon macro. The CEO macro sets the search list to include :UTIL:CEO_DIR; then chains to the CEO control program, CEO_CP.PR. By chaining, the macro ensures that when the user leaves the CEO system, he/she will be logged off the system (without returning to the CLI). Chaining also minimizes the number of processes on the system. To choose this macro, edit the user logon macro to end with

```
:UTIL:CEO_DIR:CEO.STARTUP.CLI
```

If, instead of the CEO system, you want a user to start in the CEO Word Processor, specify a different CEO–supplied macro. End the user logon macro with

```
:UTIL:CEO_DIR:CEO.WP.STARTUP.CLI
```

To select the default IPC file, press NEW LINE. Otherwise, type Y. (You will need to create an initial IPC file for every profile, since there is no default value provided for this PREDITOR question.) PREDITOR will ask for the new filename of 0–63 characters. You must type the full pathname from the root, for example,

:UTIL:LOGON_CENTRAL.CLI

Program [default] change?

The system will execute this program for the user when he or she logs on or submits batch jobs. If the user is allowed no sons (asked later), he or she will be restricted to this program, which may be just what you want.

The default program, :CLI.PR, is a good general–purpose choice. The CLI allows users to access text editors and build programs in all Data General languages. It also allows users to execute other programs like BASIC. Take the default unless you want a program other than :CLI.PR to run automatically.

16–Bit and 32–Bit CLIs

You have a choice of CLIs. Data General ships both the standard 16–bit CLI and a 32–bit CLI, which has more processing power but uses more memory.

For most operations, the two CLIs appear to be identical. But for some tasks, the 32–bit CLI has major advantages. The 32–bit CLI has far larger stack space, which lets users run macros that call themselves recursively with little fear of exhausting CLI memory. The 32–bit CLI also has a command history feature, similar to the UNIX® Berkeley C shell history feature. This history feature offers convenient access to previous commands typed; it can save time and help ease the transition to the CLI if your programmers have UNIX experience.

On the other hand, the 32–bit CLI consumes more memory than the 16–bit CLI. If your system has limited memory or is near its memory capacity, or if it has reached capacity and encountered memory contention, you should continue to use the 16–bit CLI for all users except, perhaps, the system operator. Also, the 16–bit CLI offers labeled diskette support via its OPERATOR and DUMP commands; the 32–bit CLI offers no such support for labeled diskettes. If you want or need to use labeled diskettes, you will have to run the 16–bit CLI.

You can specify the 32–bit CLI for some users and the 16–bit CLI for other users. There is some overhead in having different CLIs running, but not as much overhead as having all users run the 32–bit CLI.

The program pathname of the 32–bit CLI is :CLI32.PR; the pathname of the 16–bit CLI is :CLI16.PR. Under AOS/VS II, PREDITOR's default program name, :CLI.PR, is a link to the 32–bit CLI, :CLI32.PR; if you accept the default answer to the *Program [default] change?* question while creating or editing a user's profile, that user will run the 32–bit CLI. Under AOS/VS, PREDITOR's default program name, :CLI.PR, is a link to the 16–bit CLI, :CLI16.PR.; the default answer to the *Program [default] change?* question while creating or editing a user's profile in AOS/VS means that user will run the 16–bit CLI.

If you want a user to use the 16–bit CLI, type Y (yes) to change the default, and then type :CLI16.PR and press NEW LINE. If you know that you want all users except the system operator to use the 16–bit CLI, you can edit the !DEFAULT! profile, specifying :CLI16.PR for the value of the *Program [default] change?* question, and use the !DEFAULT! profile to create other user profiles.

Possibly you may not want the initial program to be the CLI at all. If you want this user to start in BASIC or some other program, type the full pathname of that program (including the .PR suffix). For BASIC, there is often a BASIC directory off the root (:) directory or in the directory :UTIL. If this is true (or will be true because you plan to install such a directory), answer Y (yes) to start the user in BASIC (or the preferred program).

There is another more versatile way to start a user in a program other than the CLI. For another program, choose CLI.PR or :CLI16.PR. Later on, edit the user's personal logon macro to execute the preferred program. Then, when the user logs on, the central logon macro will call the :UDD:username:LOGON.CLI macro, and this macro will execute the desired program. This approach has the advantage of allowing the user to log on even if the preferred program isn't available; the user will log on the CLI and receive a *File does not exist* message.

Create without block [default]?

A value of Yes (Y) to this question allows the user to have at least two processes running concurrently. By default, the creating (father) process is blocked when it executes the son; this means that the father is eligible to be swapped, which may speed up the system. If the user needs Data Generals SWAT® debugger (for FORTRAN 77, PL/I, COBOL, C, or PASCAL programs), he or she must have this privilege.

CEO users must have this privilege (or they will not be able to execute CEO), and programmers often need it. Other users usually don't need this privilege.

Use IPC [default]?

IPC privileges allow a person to use IPC calls, available in assembly language and some higher level languages. IPC privileges are needed wherever two or more active processes must communicate. For communication between processes to work, a user may also need the Create Without Block privilege, if two or more of that user's processes are to use IPC.

CEO users must have this privilege; most other people don't need it.

Use console [default]?

A user must have this privilege to log on a terminal under EXEC. Without it, users can submit batch jobs via a card reader, but do little else. Almost all users need this privilege.

Use batch [default]?

A user must have this privilege to submit batch jobs via CLI QBATCH or QSUBMIT commands or via a card reader. Depending on your system, you may or may not want to encourage batch jobs.

Use virtual console [default]?

This question is meaningful if your system will run the XODIAC networking system or if your system includes terminals connected by means of a "soft" controller. These terminals include NVT (Novell™ Virtual Terminal), PCVT, VTA, PAD, or TELNET. MV Data Center Manager, a separate product, also needs this privilege. A Yes (Y) value enables the user to log on your system from a virtual terminal or use the XODIAC loop–back feature.

If this user has (or will have) Superuser privilege, he or she should not have this privilege or the next privilege. A superuser with either network privilege can explore the entire system from a terminal on a remote system.

Access local resources from remote machines [default]?

This question is only meaningful if your system will run the XODIAC networking system or other networking program, like UFTAM or ftp, from another system. A Yes (Y) value allows a remote user to access files and devices like tapes and printers on your system. This is different from being able to log on from a virtual terminal or a terminal connected to a soft controller, as covered in the previous question. For remote resource access, the user must have a profile with the same username and password encrypted or unencrypted (but not necessarily the same privileges) on both systems. Details are in the manual *Managing and Operating the XODIAC™ Network Management System*.

Change password [default]?

In general, users should be able to change their own passwords, using a Y (Yes) value. If this is a GUEST profile to allow guests to use your system, the value should be No (N) to prevent a guest from changing the password and barring other guests from the system. If this value is No, the only way to change the password is with PREDITOR.

Unlimited sons [default]?

A user who can create unlimited son processes has the potential for dominating the system. Each process requires some CPU time and disk I/O. So far as possible, it is a good idea to minimize the number of processes. Generally, this value should be No.

Sons [default] change?

To use CEO Revision 3.20 and later effectively, a user needs 12 sons. If the user imports documents from other computer systems, or uses PRESENT with CEO, the user needs 15 sons.

BASIC and clerical data entry users need fewer sons, 1 or 2. To limit a user to the program you specified for them via the *Program* question, choose 0 sons. The default, 3, gives some leeway for the average non–CEO user.

To take the default, press NEW LINE. To change it, type Y. PREDITOR will say New (0–1023), and you will type the new number and press NEW LINE.

Change priority [default]?

Processes compete for CPU time, and processes of the same type with higher priority (closer to 0) get preference. But it is simpler and often better if all processes of the same type have the same priority. So generally processes should retain their initial priority.

To take the default, press NEW LINE. To change it, type Y and press NEW LINE.

Change type [default]?

Processes can run as one of three types: resident (always in main memory), pre–emptible (generally in main memory, but swappable if blocked), and swappable. Resident and pre–emptible always have priority over swappable.

Swappable is the most common type. It is the default for user processes. Resident is quite rare; often it used only for the operating system peripheral (device) manager and the operating system itself. If a process can change type, it can become resident, and slow the system. So unless you know that a process must be able to change its type, this value should be No. To take the default, press NEW LINE; to change it, type Y (Yes), the new value, and press NEW LINE.

Change username [default]?

This privilege lets a process create a son with a different username. The ability to change username allows the user to become any other user, with OWARE access to that user's files. With the username OP, the user can issue EXEC commands (allowing that user to terminate user processes or change priority or type). Also, any user with this privilege can't be traced; he or she can try to break security under any username.

For user files to remain secure and logging to be useful, every user on your system should have his/her unique username, and be unable to change it. Unless you know a process must be able to change its username, this value should be No. If you want a secure system, do not give users this privilege. To take the default, press NEW LINE; to change it, type Y (Yes), the new value, and press NEW LINE.

Access devices [default]?

This privilege allows a process to bypass operating system safeguards and define devices (via call ?IDEF), access the devices, and wire pages via privileged hardware I/O instructions. For the privilege to work, the user also needs the *Change Type* privilege (earlier) to create a resident process. An AOS/VS II system manager of a system with an MRC needs this privilege to run the Runtime Configuration Manager, which permits resetting switched routes on the MRC.

Do not give this privilege unless the user is a system manager of an AOS/VS II system with an MRC or a trusted system programmer who needs it to write or debug device drivers. Network and DG/SNA processes need it, but they can be given it from the master CLI (via the PROCESS command with /ACCESSDEVICES switch). Do not give this privilege to users if you want a secure system. To take the default, press NEW LINE; to change it, type Y, the new value, and press NEW LINE.

Superuser [default]?

This privilege allows a user process to bypass all file access control lists (ACLs) and execute, read, modify, or delete any file on the system. Only people who must bypass ACLs (to create profiles or do backups, for example) should have Superuser privilege. The master CLI needs the Superuser privilege to control the system; but most other users do not need it.

Superusers can run PREDITOR to change their own profiles, and they can learn other users passwords from profile files. The Superuser privilege, along with the Change Username privilege, is the privilege most threatening to system security. Don't give the Superuser privilege to users if you want a secure system. The value should be No. To take the default, press NEW LINE; to change it, type Y and press NEW LINE.

Superprocess [default]?

This privilege allows a user process to issue process control commands against any process. It can block a process, change process priority, become resident, or terminate any process including the master CLI, which would bring down the entire system. Unless you know that a user needs the Superprocess privilege, the value should be No. Do not give the Superprocess privilege to users if you want a secure system. To take the default, press NEW LINE; to change it, type Y and press NEW LINE.

System Manager privilege [default]?

This privilege allows the user to initialize and release job processors (relevant only with a computer that has more than one job processor), to create and delete process classes and logical processors, and to change the locality of other users processes. System Manager privilege also allows a user process to change the system date, time, ID (SYSID), bias factor, set default device characteristics, clear a device owned by another process, and override CHAR/NRM if issued from PID 2. Also, a user with this privilege can start or stop the system log (SYSLOG), start or stop Superuser or CON0 logging, and issue CONTROL @EXEC commands as though he or she were PID 2. This privilege has a significant impact on security.

Use of classes and privileged system calls can affect the performance and security of your system. The master CLI issues all commands that require the System Manager privilege; do not give this privilege to any other user unless he or she really needs it. (For example, a user needs it to run the optional Class Assignment and Scheduling Package (CLASP)).

To take the default, press NEW LINE. To change it, type Y and press NEW LINE.

Modem [default]?

If you want this user to be able to log on via a modem, this value should be Yes. Superusers should never be able to use a modem, for the two privileges allow users to explore the entire system from their own home or wherever a remote terminal is placed. If a person must be a superuser, give him or her two profiles — one with Superuser privilege and one with Modem privilege. To take the default, press NEW LINE; to change it, type Y and press NEW LINE.

Change address space type [default]?

This privilege allows the user to execute 16–bit programs from a 32–bit program (like the 32–bit CLI) and 32–bit programs from a 16–bit program (like the 16–bit CLI). This value should be Yes. To take the default, press NEW LINE; to change it, type Y, the new value, and press NEW LINE.

Change working set limit [default]?

This privilege allows the user to run programs that change the system default working set limit. By default, the operating system adjusts the working set parameters dynamically, based on the process type, page fault history, and general system overhead.

Certain application programs, under unusual conditions, may need to change the default working set limits. The CONTEST confidence test, described later in the manual, does need to make this change. For the user process that runs such programs, the value must be Yes. Also if you want to restrict a process working set or give it a generous minimum working set (later questions), the value must be Yes; otherwise, the profile will be inconsistent and won't work. In other cases, the value should be No. To take the default, press NEW LINE; to change it, type Y, the new value, and press NEW LINE.

Priority [default]

All user processes are created as swappable processes, with the priority given here. We recommend the value 2, the original default. If a user process needs to change priority or type — of itself or its sons — you can give it the pertinent change privilege above. For the value 2, press NEW LINE (or type 2) and skip to the next question.

If you don't want the default priority offered, type Y, the new value, and press NEW LINE.

Let system assign?

This lets you create the process with its father's (EXEC's) priority. This is a bad idea, since EXEC is not a typical process and some sites give it a high priority for better queuing performance. If you want the user process created with EXEC's priority, answer Y to this question.

If you answer N or press NEW LINE for this question, PREDITOR will respond *New (1–511)*. Type the new value and press NEW LINE.

You can specify any priority, range 1–511. If you specify 1, 2, or 3, the user process will be created as a group 2 (heuristically scheduled) process, with first, second, or third priority within group 2. Priority 2, the original default, gives the same results as the default in earlier AOS/VS revisions.

If you want the process to have the highest priority in group 2, specify 1. If you want the process to have the lowest priority in group 2, specify 3.

If you want this user process to be created as a group 1 or group 3 (round–robin scheduled) process, specify a priority that is unique within its group range. If you took the VSGEN defaults for process groups, any priority from 4 through 255 is unique and will produce a group 1 process. Any priority from 259 through 511 is unique and will produce a group 3 process.

Max qpriority [default] change?

Users type Q–series commands (QPRINT, QBATCH, etc.) to print files and submit batch jobs. Priority 0 is the default and highest priority for these. If all users have the same priority, they will receive equal treatment on their Q–series jobs. Generally, unless you want to prioritize different users Q–series requests, keep the same value, usually 0. To take the default, press NEW LINE. To change it, type Y; PREDITOR will respond New(0–255), and you'll type the new value and press NEW LINE.

Disk quota [default] change?

Your response to this question sets the limit on the size of the user directory that PREDITOR will create and that the system will maintain for this user process. (This PREDITOR–assigned limit doesn't affect users' space allocation in the CEO system.)

A good general–purpose amount of disk space is 15,000 blocks. If this user process will serve the CEO system, guest users, or other infrequent users, you might want to specify less space (perhaps the original default, 500). If this user process will serve many people (perhaps data entry personnel or students), you might want to specify a larger figure (e.g., 100,000).

If this user process will deal with a large database and its directory will contain the database(s), you might want to allot an entire single– or multiple–disk LDU to it. A model 6061 disk contains about 370,000 blocks; a model 6122 disk contains about 540,000 blocks; a model 6236 disk contains about 690,000 blocks; and a model 6239 disk contains about 1,150,000 blocks.

To take the default, press NEW LINE. To change the space quota, type Y and press NEW LINE. PREDITOR will then ask New(0–2147483647): and you will type the new quota. Don't use commas in your answer. For more on disk space quotas, see the section "Disk Space Control," later in this chapter.

Logical address space – batch [–1 system default] change?

This question and the next can override system limits and set a new maximum number of memory pages that this user process and its sons can address. (Do not confuse *logical address space* with *working set size*.) The Link utility sets the logical address space for a program. The operating system checks this value, and will not create a process if the program's value exceeds the value you set here. For this reason, there is no reason to change this value.

The answer to this question should be the value "–1 system default." This value gives the user process the same limit as its father, EXEC (which in turn gets its limit from its father, the master CLI). If you want to specify a different value, type Y and press NEW LINE. PREDITOR will then ask *Let system assign?* This means the same thing as "–1 system default," so if you press NEW LINE you receive the original default. If you want to change the value to something other than "–1 system default," answer N and press NEW LINE to the *Let system assign?* question. PREDITOR then displays a range of values from which you can choose, for example, New (1–262144). Type Y, the new value, which must be a multiple of 2–Kbyte pages, and press NEW LINE.

Logical address space – non–batch [–1 system default] change?

This question allows you to set a nondefault page limit as above, but for interactive use. The same constraints and dialog as shown earlier apply.

Minimum working set size – batch [–1 system default] change?

This series of questions allows you to set nondefault lower and upper working set limits for batch and interactive processes running under this process.

Normally, the operating system adjusts the working set parameters dynamically, based on process type, page fault history, and general system overhead. These questions allow you to keep the working set within specific limits.

Defining a process's working set can be useful if memory contention is heavy, but it can also cause thrashing. So we recommend that you take the default to these questions and rely on the operating system adjustments.

A process working set does not necessarily relate to the size of its logical address space. In fact, a larger process may require fewer pages than a smaller one if it localizes its references. If you do set working set limits, take care not to set the maximum too low, since this increases paging I/O and elapsed processing time. Also note that setting the minimum working set size too low can degrade performance throughout the system by increasing page faults.

For this question, generally, the value should be "– 1 system default." If you want to change this value, type Y and press NEW LINE. PREDITOR will then ask "Let system assign?" If you press NEW LINE, you receive the system default. If you reply N and press NEW LINE, PREDITOR will ask New (40–32767), and you will type the new value and press NEW LINE.

Maximum working set size – batch [–1 system default] change?

This question allows you to specify a nondefault maximum working set limit for batch processes under this user process. The same constraints as above apply.

Minimum working set size – non–batch [–1 system default] change?

Your response to this question allows you to specify a nondefault minimum working set limit for interactive processes under this user process. The same constraints as above apply.

Maximum working set size – non–batch [–1 system default] change?

This question asks you to specify a nondefault maximum working set limit for interactive processes under this user process. The same constraints as above apply.

Default user locality – non–batch [default] change?

This question and the following questions about locality are important only if you want to use class scheduling on your system. You can create and implement classes with the optional Class Assignment and Scheduling Package (CLASP), described in the manual *Class Assignment and Scheduling Package*; or you can write a program to do it through system calls.

PREDITOR asks first about interactive (non–batch) processes; then about batch processes, so that you can treat the two differently.

Generally, even if you do use classes, it is a good idea to retain the original default value for this question and for the related batch question. You can allow the user access to other localities in the next question. (If you want to make sure the user comes up in a nondefault user locality, you can specify that locality number.) To take the default, press NEW LINE. To change it, type the new value and press NEW LINE.

Use other localities – non–batch [default]?

If you don't want to use classes, ignore this question and press NEW LINE. If you do want to use classes, decide whether or not you want this user to be able to use other localities for interactive processes. Users can change locality, if privileged, with the LOCALITY command or PROCESS command with the /LOCALITY switch. You may want to start the user in the original default locality (0) and change him or her to a different locality (the range is 0 – 15), perhaps in the user's logon macro. If you do allow other localities, the user may be able to enter a privileged class and obtain more than his/her share of processing time, depending on your class definitions.

To take the default, press NEW LINE; to change it, type the desired answer and press NEW LINE. After a No answer, PREDITOR skips the next question. After a Yes answer, PREDITOR asks

User locality – non–batch [default] change?

The user will be able to change locality to any locality you specify here. Respond with the numbers of all localities you want the user to have for interactive processes, or press NEW LINE to prevent the user from changing locality. Separate numbers with spaces; for example, type

1 2 3 ↵

Default user locality – batch [default] change?

This question and the next are like the previous ones except that they apply to batch processes. As above, generally accept the defaults unless you are setting up your own scheduling scheme. To take the default, press NEW LINE. To change it, type the new value and press NEW LINE.

Use other localities – batch [default]?

This question lets you choose whether the user can use other localities for batch processes. If you want the default, press NEW LINE. Otherwise, type the desired answer and press NEW LINE. After a Yes answer, PREDITOR asks

User locality – batch [default] change?

The user will be able to change the locality of batch processes to any locality you specify here. Respond with the numbers of all localities you want the user to have for batch processes, or press NEW LINE to prevent the user from changing locality. Separate numbers with spaces; for example, type

1 2 3 ↵

User comment [default] change? (Y or NL)?

This is your opportunity to add comments about the user for whom you have created this profile: full name, date, etc. If you do not want to comment on the profile, press NEW LINE. To enter or change a comment, type Y and press NEW LINE; then type your comments; for example,

User comment [default] change? (Y or NL)? Y ↵

New (0–79 chars): JACK ARMSTRONG. GIVEN MODEM 22 NOV 93 ↵

Command:

You've finished the CREATE/EDIT session; and the new/edited profile is ready for its user. An EDIT example follows the next section.

Disk Space Control

PREDITOR's disk quota question asks you to set the maximum amount of disk space for a user not in the CEO system. This space includes all files and subordinate directories the user may create; it also includes space requirements of all processes run under this user process.

The quota limit should be large enough to allow the user (process) to work effectively. High quotas help prevent users from running out of space — an error situation that requires the system operator to edit profiles and provide more space.

On any LDU, you can allot users more disk space than actually exists. For example, if the LDU contains 370,000 blocks, you can give each of 40 users a quota of 10,000 blocks. This practice is called oversubscribing an LDU. It is possible because the quota is not actually reserved for each user; it is a theoretical limit, but is not guaranteed to the user.

The advantage of oversubscribing is that it allows generous disk quotas, which prevents users from running out of disk space and requiring someone to edit their profiles (to increase their disk quotas).

There are however several disadvantages:

- Users can become careless and saturate the LDU with unimportant files;
- The LDU can become nearly filled with files, which will slow the system;
- The LDU can get entirely filled, which will cause all users to get error messages when they try to create files, even though their individual disk quotas haven't been reached.

Be especially careful about oversubscribing an LDU that contains the SWAP and PAGE directories. If an LDU contains the SWAP and PAGE directories, and it fills up, the system will panic.

To check the actual amount of free space on an LDU, use the CLI command SPACE, with the LDU master directory name as an argument; for example, type

Su) SPACE : ↵

and press NEW LINE. If 10% or less of the total LDU space remains, you may want to free some space by moving files to another LDU or dumping files, then deleting them from this LDU; or by telling users to clean up their directories.

When you edit a profile, don't reduce the disk quota below the amount of space the user is actually using. If you do this, the user won't be able to log on. When you want to reduce a disk quota, use the command line

Su) SPACE :UDD:username ↵

before running PREDITOR to see how much space is occupied; this is the minimum disk quota you should specify to PREDITOR for this user.

EDIT Dialog and Example

The following example shows several editing changes in the profile of a user named Jack.

) X PREDITOR ↵

User Profile Editor Rev n date time

Command: EDIT ↵

Username: JACK ↵

Password change? (Y or NL) ↵

Initial IPC file [:UDD:JACK:LOGON.CLI] change? (Y or NL) ↵

Program [:CLI.PR] change? (Y or NL) ↵

Create without block [No]? (Y, N, or NL) Y ↵

Use IPC [No]? (Y, N, or NL) Y ↵

Use console [No]? (Y, N, or NL) ↵

Use batch [Yes]? (Y, N, or NL) ↵

Use virtual console [Yes]? (Y, N, or NL) ↵

Access local resources from remote machines [Yes]? (Y, N, or NL) ↵

Change password [Yes]? (Y, N, or NL) ↵

Unlimited sons [No]? (Y, N, or NL) ↵

Sons [3] change? (Y or NL) Y ↵

New (0–1023): 12 ↵

Change priority [No]? (Y, N, or NL) ↵

Change type [No]? (Y, N, or NL) ↵

Change username [No]? (Y, N, or NL) ↵

Access devices [No]? (Y, N, or NL) ↵

Superuser [No]? (Y, N, or NL) ↵

Superprocess [No]? (Y, N, or NL) ↵

System manager privilege [No]? (Y, N, or NL) ↵

Modem [No]? (Y, N, or NL) Y ↵

Change address space type [Yes]? (Y, N, or NL) ↵

Change working set limit [No]? (Y, N, or NL) ↵

Priority [2] change (Y or NL) ↵

Max qpriority [0] change (Y or NL) ↵

Disk quota [15000] change (Y or NL) Y ↵

New (0–2147483647): 20000 ↵

Logical address space–batch [–1 system default] change? (Y or NL) ↵

Logical address space–non–batch [–1 system default] change? (Y or NL) ↵

Minimum working set size–batch [–1 system default] change? (Y or NL) ↵

Maximum working set size–batch [–1 system default] change? (Y or NL) ↵

Minimum working set size–non–batch [–1 system default] change? (Y or NL) ↵

Maximum working set size – non–batch [–1 system default] change?(Y or NL) ↵

Default user locality – non–batch [0] change? (Y, N, or NL) ↵

Use other localities – non–batch [No]? (Y, N, or NL) ↵

Default user locality – batch [0] change? (Y, N, or NL) ↵

Use other localities – batch [No]? (Y, N, or NL) ↵

User comment [JACK ARMSTRONG 22 NOV 91] change? (Y or NL)? Y ↵

New (0–79 CHARS): JACK ARMSTRONG. GIVEN MODEM 22 NOV 93 ↵

Command: BYE ↵

Terminating date time

)

DELETE Command: Deleting a User Profile

PREDITOR's DELETE command deletes a user profile which prevents the user from using the system. Optionally, it also deletes the user directory.

PREDITOR asks for a username, then asks for confirmation of the profile delete, and then asks if you want to delete the user directory. If you reply Y and press NEW LINE, PREDITOR deletes the user directory and all its subordinate directories. You can use a template for the username. If you do, PREDITOR will prompt you to delete matching users profiles and directories, one user at a time.

You can use this command whenever you want to terminate someone's ability to use the system — perhaps because this person has left the organization or is no longer a suitable time–sharing user. For the profile (:UPD:username) and/or selected files (template :UDD:username:#). Note, however, that PREDITOR does not resolve links. For example, if a user's directory is on a separate disk (like UDD1), and username is a link in UDD, PREDITOR will delete the link, not the directories on UDD1.

If you care about security, you should dump the directory and then delete it. Dumping the files to tape will prevent anyone from creating a profile with the same username, which would give the new user OWARE access to the old users files.

For information on deleting a CEO user's documents, read the manual *Managing the CEO® System*.

DELETE Dialog and Example

The following examples shows how you would go about deleting the user profile and directory of a user named Simon.

Command: DELETE ↵

Username: SIMON ↵

Delete user SIMON? (Y or N) Y ↵

User deleted

Delete user directory :UDD:SIMON? (Y or N) Y ↵

User directory deleted

Command:

EDIT Command: Editing an Existing Profile

See the CREATE command.

HELP Command: Getting Help

The HELP command gives summary information on all PREDITOR commands.

HELP Dialog and Example

Command: HELP ↵

The legal commands are:

Bye
Create
Delete
Edit
Help
List
Question (turn questions on or off)
Rename (rename a profile and user directory)
Use (use another profile as !DEFAULT!)

Command:

LIST Command: Displaying Values in a Profile

The LIST command asks for a username; then displays all the current values in a profile. This command is useful when you want to know every value in a user profile. You can list the profiles for all users by typing + at the username prompt; the values for all user profiles will scroll on the screen. If you want to generate a hardcopy file of all user profiles on your system, execute PREDITOR with the /L=filename switch. For example

```
) X PREDITOR/L=USER_PROFILES ↵
```

At the *Command* prompt, type LIST, and at the *Username* prompt, type +. PREDITOR will list all the values for all user profiles in the file you specify. You can then print the file in order to review user privileges.

LIST Dialog and Example

The following example lists the values in the user profile of the user named Jack.

Command: LIST ↵

Username: JACK ↵

Encrypt password [Yes]

Initial IPC file [:UDD:JACK:LOGON.CLI]

Program [CLI.PR]

Create without block [Yes]

Use IPC [No]

Use console [Yes]

Use batch [Yes]

Use virtual console [Yes]

Access local resources from remote machines [Yes]

Change password [Yes]

Unlimited sons [No]

Sons [12]

Change priority [No]

Change type [No]

Change username [No]

Access devices [No]

Superuser [No]

Superprocess [No]

System manager privilege [No] Modem [Yes]

Change address space type [Yes]

Change working set limit [No]

Priority [2]

Max qpriority [0]

Disk quota [20000]

Logical address space – batch [–1 system default]

Logical address space – non–batch [–1 system default]

Minimum working set size – batch [–1 system default]

Maximum working set size – batch [–1 system default]

Minimum working set size – non–batch [–1 system default]

Maximum working set size – non–batch [–1 system default]

Default user locality – non–batch [0]

Use other localities – non–batch [No]

Default user locality – batch [0]

Use other localities – batch [No]

User comment [JACK ARMSTRONG. GIVEN MODEM 22 NOV 93]

Command:

QUESTION Command: Suppressing or Restoring Questions

The QUESTION command turns off display of any or all PREDITOR questions. The questions you suppress remain suppressed until you re–issue the QUESTION command or leave PREDITOR.

When you type QUESTION, PREDITOR displays each question individually, in the CREATE/EDIT order. Press NEW LINE if you want PREDITOR to ask the question; type N and press NEW LINE to suppress the question. Instead of the usual [default] value, PREDITOR displays [Y], to indicate that by default it will display the question.

The QUESTION command is handy when you want to change only a few parameters for many users on the system. You can suppress all the irrelevant questions with this command; then edit all the profiles you want quickly.

CAUTION: Don't create a new profile with questions suppressed; such a profile may be unusable. Suppress the questions only when editing profiles.

QUESTION Dialog and Example

In the following example, the person running PREDITOR wants to edit only the *Create without block*, *Disk quota*, and *User comment* values for users LEE and F77. The QUESTION command helps speed this up. First the person running PREDITOR uses the QUESTION command to list all the PREDITOR questions, and chooses which values (or privileges) they want.

Command: QUESTION ↵

Password [Y]? (Y or NL) N ↵

Initial IPC file [Y]? (Y or NL) N ↵

Program [Y]? (Y or NL) N ↵

Create without block [Y]? (Y, N, or NL) ↵

Use IPC [N]? (Y, N, or NL) N ↵

Use console [Y]? (Y, N, or NL) N ↵

Use batch [Y]? (Y, N, or NL) N ↵

Use virtual console [Y]? (Y, N, or NL) N ↵

Access local resources from remote machines [Y]? (Y, N, or NL) N ↵

Change password [Y]? (Y, N, or NL) N ↵

Unlimited sons [Y]? (Y, N, or NL) N ↵

Sons [Y]? (Y, N, or NL) N ↵

Change priority [Y]? (Y, N, or NL) N ↵

Change type [Y]? (Y, N, or NL) N ↵

Change username [Y]? (Y, N, or NL) N ↵

Access devices [Y]? (Y, N, or NL) N ↵

Superuser [Y]? (Y, N, or NL) N ↵

Superprocess [Y]? (Y, N, or NL) N ↵

System manager privilege [N]? (Y, N, or NL) ↵

Modem [Y]? (Y, N, or NL) N ↵

Change address space type [Y]? (Y, N, or NL) N ↵

Change working set limit [Y]? (Y, N, or NL) N ↵

Priority [Y]? (Y, N, or NL) N ↵

Max qpriority [Y]? (Y, N, or NL) N ↵

Disk quota [Y]? (Y, N, or NL) ↵

Logical address space – batch [Y] (Y, N, or NL) N ↵

Logical address space – non–batch [Y] (Y, N, or NL) N ↵

Minimum working set size – batch [Y] (Y, N, or NL) N ↵

Maximum working set size – batch [Y] (Y, N, or NL) N ↵

Minimum working set size – non–batch [Y] (Y, N, or NL) N ↵

Maximum working set size – non–batch [Y] (Y, N, or NL) N ↵

Default user locality – non–batch [Y]? (Y, N, or NL) N ↵

Use other localities – non–batch [N]? (Y, N, or NL) ↵

User locality – non–batch [Y]? (Y, N, or NL) N ↵

Default user locality – batch [Y]? (Y, N, or NL) N ↵

Use other localities – batch [N]? (Y, N, or NL) ↵

User locality – batch [Y]? (Y, N, or NL) N ↵

User comment [Y]? (Y, N, or NL)? ↵

Now the person running PREDITOR can quickly edit the two profiles for LEE and F77, answering only the selected questions.

Command: E ↵

Username: LEE ↵

Create without block [No]? (Y, N, or NL) Y ↵

Disk quota [15000] change? (Y or NL) Y ↵

New (0–2147483647): 25000 ↵

User comment [LEE WILLIAMS 30 NOV 84] (Y or NL) change? Y ↵

New (0–79 CHARS): LEE WILLIAMS, 25,000 BLOCKS, 22 NOV 93 ↵

Command: E ↵

Username: F77 ↵

Create without block [Yes]? (Y, N, or NL) ↵

Disk quota [50000] change? (Y or NL) ↵

New (0–2147483647): 200000 ↵

User comment [F77 PROJECT 30 NOV 87] change? (Y or NL)? Y ↵

New (0–79 CHARS): F77 PROJECT UP TO 200,000 BLOCKS 22 NOV 93 ↵

Command: BYE ↵

RENAME Command: Renaming a Profile

The RENAME command renames a user profile and its associated user directory. Within the profile, only the username changes; the password and all other values remain unchanged.

The RENAME command also changes the ACL for the user directory to new–username,OWARE — giving the new username all access privileges to the directory. This change may cause problems if the user is logged on under the old name, so rename a profile only when the user is not logged on under the original username.

The RENAME command does not change the ACLs of files and subdirectories. All files and subordinate directories within the user directory retain their old ACLs preventing access by the new username. After renaming a profile, either you or the user should update these ACLs via the CLI using the following format.

```
DIR :UDD:username

ACL/V # new–username,OWARE
```

If you (not the user) change the ACLs, you will need to turn Superuser on first.

You might use the RENAME command if a user didn't like his or her assigned username, or if a more explicit or descriptive username was desirable. For a network user, renaming a profile may prevent access to other hosts (explained earlier under "CREATE/EDIT Questions").

RENAME Dialog and Example

First the person running PREDITOR uses the RENAME command to change the username in the profile of the user named Sally.

Command: RENAME ↵

Username: SALLY ↵

New username: SAL ↵

Command: B ↵

Terminating date time

Then he or she turns on Superuser privilege and changes the ACLs of all the files and subdirectories contained in the user directory (now named) SAL.

) SUPERUSER ON ↵

Su) DIR :UDD:SAL ↵

Su) ACL/V # SAL, OWARE ↵

... (Verification) ...

Su) SUPERUSER OFF ↵

) DIR/I ↵

USE Command: Using Another Profile as !DEFAULT!

PREDITOR displays the values in its !DEFAULT! profile as defaults for CREATE and EDIT questions. The USE command tells it to use the values in another profile for these defaults.

During a PREDITOR session, you can edit !DEFAULT! to change the defaults — but because the !DEFAULT! profile is internal to PREDITOR, these changes remain only until PREDITOR terminates. User profiles, which are files in :UPD, on the other hand, remain stable. So when you create a new profile that will be similar to an existing one, you can use the existing profile and change only the username, password, and possibly a few other values. PREDITOR continues to use the values in the existing profile until it terminates or until you tell it to use another profile.

The USE command doesn't affect the existing profile; the values in the existing profile serve as defaults.

USE Dialog and Example

This example shows how to use an existing (Jack's) profile to help create a profile for a new user named Barbara.

) X PREDITOR ↵

AOS/VS II User Profile Editor Rev n date time

Command: USE ↵

Username: JACK ↵

... (PREDITOR displays values in JACK's profile) ...

Command: C ↵

Username: BARBARA ↵

Password change? (Y or NL) Y ↵

New (6–15 CHARS): BARBARA ↵

Encrypt password [Yes]? (Y, N, or NL) ↵

Initial IPC file [:UDD:JACK:LOGON.CLI] change? (Y or NL) Y ↵

New (0–63 chars): :UDD:BARBARA:LOGON.CLI ↵

Program [:CLI.PR] change? (Y or NL) ↵

Create without block [Yes]? (Y, N, or NL) N ↵

Use IPC [No]? (Y, N, or NL) ↵

Use console [Yes]? (Y, N, or NL) ↵

Use batch [Yes]? (Y, N, or NL) ↵

Use virtual console [Yes]? (Y, N, or NL) ↵

Access local resources from remote machines [Yes]? (Y, N, or NL) ↵

Change password [Yes]? (Y, N, or NL) ↵

Unlimited sons [No]? (Y, N, or NL) ↵

Sons [12] change? (Y or NL) ↵

Change priority [No]? (Y, N, or NL) ↵

Change type [No]? (Y, N, or NL) ↵

Change username [No]? (Y, N, or NL) ↵

Access devices [No]? (Y, N, or NL) ↵

Superuser [No]? (Y, N, or NL) ↵

Superprocess [No]? (Y, N, or NL) ↵

System manager privilege [No] change? (Y, N, or NL) ↵

Modem [No]? (Y, N, or NL) Y ↵

Change address space type [Yes]? (Y, N, or NL) ↵

Change working set limit [No]? (Y, N, or NL) ↵

Priority [2] change? (Y or NL) ↵

Max qpriority [0] change? (Y or NL) ↵

Disk quota [20000] change? (Y or NL) ↵

Logical address space – batch [–1 system default] change? (Y or NL) ↵

Logical address space–non–batch [–1 system default] change? (Y or NL) ↵

Minimum working set size – batch [–1 system default] change? (Y or NL) ↵

Maximum working set size – batch [–1 system default] change? (Y or NL) ↵

Minimum working set size–non–batch [–1 system default] change? (Y or NL) ↵

Maximum working set size–non–batch [–1 system default] change? (Y or NL) ↵

Default user locality – non–batch [0] change? (Y or NL) ↵

Use other localities – non–batch [No]? (Y, N, or NL) ↵

Default user locality – batch [0] change? (Y or NL) ↵

Use other localities – batch [No]? (Y, N, or NL) ↵

User comment [JACK ARMSTRONG 9 MAR 84] change? (Y or NL)? Y ↵

NEW (0–79 CHARS): BARBARA CLEAVES, 22 NOV 93 ↵

After Jack's user profile values have been listed, PREDITOR will once again display the *Command* prompt. Type

Command: CREATE ↵

Username: BARBARA ↵

and PREDITOR will list all the values taken from Jack's user profile one at a time. The only value that you will have to change is *Password*. For all the other values, you can accept the default value by pressing NEW LINE.

End of Chapter

Chapter 3
Managing User Processes with EXEC

This chapter describes how to use the EXEC program to manage a multiuser environment on an AOS/VS or AOS/VS II system.

Read this chapter to learn about managing user logon, and managing printing, backup, batch, and network processing queues.

The major sections in this chapter are

- What EXEC Does
- Managing EXEC
- Pertinent CLI Commands
- EXEC Command Overview
- Managing User Logon
- Managing Queues and Devices
- Managing Print Processing
- Managing Batch Processing
- Managing Communications and Network Queue Processing
- Managing Mount Processing
- EXEC Commands

There are a number of interfaces to EXEC, including the CLI, the System Management Interface (SMI), and CEO. This chapter describes the CLI interface to EXEC's system management functions. If you are managing EXEC through an interface other than CLI, use this chapter to supplement the information in the manual appropriate for that interface.

What EXEC Does

The EXEC program, located in :UTIL, supervises the operating system multiuser environment. When it is running, EXEC performs the following tasks:

- Manages user logon and logoff. When a user tries to log on, EXEC checks for a valid profile. Then if the username and password that the user types match those in a profile, EXEC creates a user process, with parameters defined in the profile, for that user. When the user logs off, EXEC terminates the user process.
- Manages queues of requests for batch, print, plot, mount, FTA, SNA, HAMLET, or user queue processing. System managers create and manipulate these queues with CONTROL @EXEC commands. Users queue, examine, and modify their requests, or jobs, with CLI Q–series commands.
- Manages EXEC cooperatives. EXEC cooperatives are programs which process jobs in EXEC queues. XLPT.PR is one example of an EXEC cooperative; it processes print jobs by formatting the data to be printed and outputting it to a printer.

EXEC commands control all these functions. Usually — after the UP.CLI macro brings it up — EXEC runs practically by itself.

Managing EXEC

All the EXEC files are in directory :UTIL. These include the

- EXEC program file, EXEC.PR;
- line printer and plotter manager program, XLPT.PR;
- batch processor program, XBAT.PR;
- tape mount and dismount manager program, XMNT.PR;
- network interface manager program, XNET.PR;
- forms control utility, FCU.PR; and
- queue cleanup program, QCMP.PR.

The system expects to find EXEC files in directory :UTIL, so don't move them.

Creating the EXEC Process

The UP.CLI macro includes the CLI PROCESS command that creates EXEC, and EXEC commands to enable terminals and start queues. You can type or print the file :UP.CLI to see them. For completeness, we will show the PROCESS command here. It is

PROCESS/DEFAULT/DIRECTORY=@/NAME=EXEC EXEC*[/SUPPRESS]*

where

PROCESS	is a CLI command that creates a new process (just as XEQ does, but the PROCESS command is more versatile).
/DEFAULT	gives the new process all the privileges of the father process. The father is nearly always the master CLI, PID 2, which has the PREDITOR privileges that EXEC needs.
/DIRECTORY=@	makes the peripheral directory (:PER, shorthand @) the home directory of the EXEC process. This is necessary because all the device entries EXEC needs are in :PER.
/NAME=EXEC	gives the EXEC process the simple process name EXEC so that the system can access it by this process name. (Note that EXEC's process name must be EXEC; other processes, such as SVTA, will not come up if EXEC has a different name.)
EXEC	is the name of the program to execute.
/SUPPRESS	tells EXEC to suppress all status messages. This switch is optional.

When EXEC starts up, it checks for another EXEC process; if an EXEC is already running, the new EXEC will terminate with an error message. When ready, EXEC displays

From Pid n : (EXEC) Revision n Ready
From EXEC hours:minutes:seconds

EXEC is ready for CONTROL @EXEC (CX) commands — many of which follow the PROCESS command in the UP.CLI macro. (The second half of this chapter contains all the EXEC commands, arranged in alphabetical order, with a full explanation of all switches, as well as how and when to use each command.)

The master CLI, PID 2, is usually EXEC's father. All the processes EXEC creates are EXEC's sons. The process hierarchy looks like Figure 3–1.

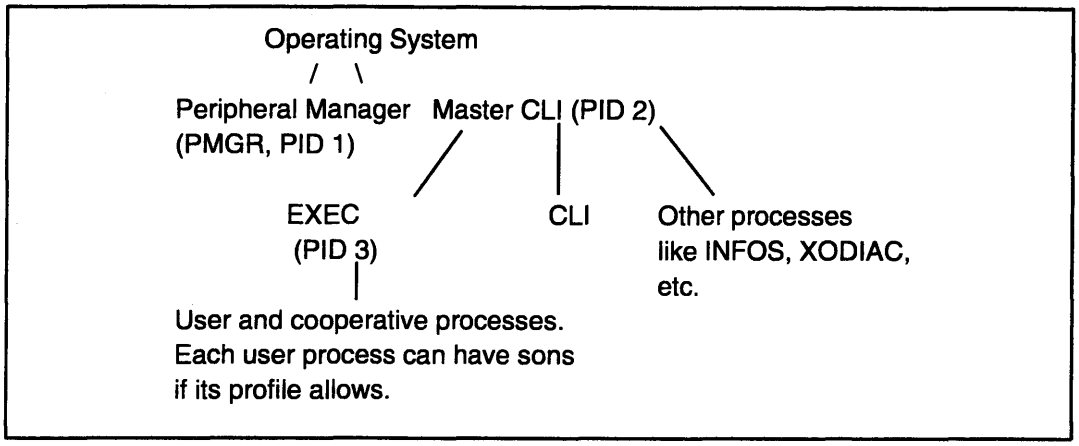


Figure 3–1 Process Hierarchy (Tree)

Monitoring EXEC Operations

There are four EXEC commands you can use to monitor ongoing EXEC operations:

- LOGGING enables you to keep a running log of EXEC;
- MESSAGE allows you to add comments to the log;
- MDUMP creates a memory dump of the EXEC process and writes it to disk;
- PROMPTS has EXEC indicate the current time in every message it sends to the system console.

All EXEC commands are described in detail, with examples, in the "EXEC Commands" section later in this chapter.

Terminating the EXEC Process

The DOWN.CLI macro includes the EXEC HALT command that terminates EXEC. Another command in DOWN.CLI — CHECKTERMS — relates to HALT, so we'll explain both. They are

```
CONTROL @EXEC HALT
CHECKTERMS
```

where

HALT	terminates the EXEC process in an orderly way.
CHECKTERMS	tells the CLI to save and display any messages that the system displays after it terminates a process.

NOTE: Normally, you would pause EXEC queues and warn users before bringing down EXEC via the DOWN macro. Halting EXEC terminates all its sons; each user process created by EXEC is a son of EXEC. Users might lose work and be upset if their processes are terminated without warning.

CLI Commands Pertinent to EXEC

While EXEC runs, you'll use some or all of the CLI commands shown in Table 3–1. See the manual *Using the CLI (AOS/VS and AOS/VS II)* for more detail.

Table 3–1 Process–Oriented CLI Commands

Command	What It Does	Example
ACL	Sets the access control list for one or more files; further described in the section "Controlling Access with ACLs" in Chapter 14.	) ACL + OP,OWARE ↵
BYE	Terminates the current CLI or SED text editor process. The process's father (from which the process was executed) get control. If this is a user process, BYE logs the user off. If there is no father, as with the master CLI, BYE starts system shutdown.	) BYE ↵
CONTROL @EXEC cmd or CX cmd	CONTROL @EXEC tells the CLI to pass cmd to EXEC. The cmd is one of the EXEC commands described in this chapter. CX invokes the macro :UTIL:CX.CLI that contains the characters CONTROL%0/% @EXEC%1–%, allowing you to type CX instead of CONTROL @EXEC.	) CONTROL @EXEC STAT ↵) CX STATUS ↵
EXECUTE	Creates a process; same as XEQ.	See XEQ.
FILESTATUS	Lists filenames in a directory. Useful switches are /AS (for assorted information) and /S (sorts filenames alphabetically). If you get an *Access denied* message, turn Superuser on and retry.	) F/AS/S :UTIL ↵

(continued)

Table 3–1 Process–Oriented CLI Commands

Command	What It Does	Example
HELP *[command]*	Gives help on CLI topics or commands. HELPB uses the BROWSE utility to view the EXEC Help files.	) HELP ↵) HELP/V ↵) HELPB ACL ↵
MOUNT DISMOUNT	MOUNT requests operator to mount a tape or multi–volume labeled tape on a tape drive. DISMOUNT asks operator to remove a tape.	) MOUNT TAPE1 & ↵ *&)* MOUNT TAPE Z280 ↵) DISMOUNT TAPE1 ↵
PROCESS	Creates a process; described in the section "Creating the EXEC Process."	) PROC program ↵
QBATCH QPRINT QSUBMIT QPLOT QSNA QFTA	These are user commands that place requests on EXEC queues. To use them, a person needs a user profile. So even if you are a system manager, you must have a profile. Chapter 5 in the *Installing* manual for your particular operating system tells how to create a privileged profile named OP.	) QBATCH X MYPROG ↵) QPRINT MYFILE ↵) QSUBMIT AFILE ↵) QPLOT THISFILE ↵) QSNA MYFILE ↵) QFTA THATFILE ↵
QDISPLAY	Describes the status of all EXEC queues. For more information, include the /V switch.	) QDISPLAY ↵) QDISPLAY/V ↵
RUNTIME *[pid]*	Describes how long a process has been running, and its CPU and I/O usage. The pid is the process ID; if you omit it, the command describes the current process.	) RUN 3 ↵) RUN ↵

(continued)

Table 3–1 Process–Oriented CLI Commands

Command	What It Does	Example
SEND user(s)	Sends a message to one or more user terminals. The user(s) can be a process ID or a template like @CON– to specify all terminals. The BROADCAST macro is preferred over using SEND @CON–.	) SEND @CON3 Log off ↵) SEND @CON– Log off ↵
TERMINATE	Terminates a process, shown in the section "Terminating the EXEC Process." The process can be a full process name like OP:EXEC, a simple process name like EXEC, or a process ID like 3.	) TERMINATE 20 ↵) TERM 18 ↵
TREE *[pid]*	Describes a process family tree: process IDs of the father, self, and son(s), if any.	) TREE 3 ↵
WHO *[pid]*	Describes the username associated with a process ID.	) WHO 4 ↵) WHO ↵
XEQ	Creates a process. Same as the EXECUTE command.	) XEQ VSGEN↵) X SED↵
XHELP or XHELPB	Displays information about all EXEC commands or a specific command. XHELPB uses the BROWSE utility to view the EXEC Help files.	) XHELP ↵) XHELP CREATE ↵) XHELPB ACCESS ↵
?	Invokes a CLI macro, ?.CLI, that contains the pseudomacro [!pids]; it describes all users on the system. The Installing manual for your operating system shows how to create this macro.	) ? ↵

(concluded)

EXEC Command Overview

EXEC commands manage user logon and batch, print, plot, mount, FTA, SNA, HAMLET, and user–created queues.

Since EXEC commands can affect the entire user community, access to these commands is initially restricted, so that only processes with System Manager privilege or processes running under the same username as EXEC (usually the username is OP), can issue EXEC commands. In this chapter, it is assumed that the person issuing EXEC commands is the system manager.

Depending on the security policies at your installation, access to some EXEC commands may be granted to other users with the EXEC ACCESS command. This command is described later in the chapter, in the "EXEC Commands" section. You can also restrict access to EXEC commands by using the 32–bit CLI LOCK/CX command, described in the manual *Using the CLI (AOS/VS and AOS/VS II)*.

Getting Help with EXEC

You can get help with EXEC commands by typing

) XHELP ↵

or

) XHELP cmd ↵ or XHELPB cmd ↵ where cmd is an EXEC command.

EXEC's help messages can be very useful when you have forgotten a command word or the correct syntax of a command. EXEC's help messages are easily available, and they can be more up to date than this manual.

Command Abbreviations

Except for XHELP, each EXEC command begins with CONTROL @EXEC or CX. As with the CLI, you can abbreviate any EXEC command to its shortest unique string. For example, instead of CX STATUS, you could type

) CX STAT ↵

Don't be afraid to try abbreviations; at worst you'll get a harmless *Command abbreviation not unique* error message.

EXEC Command Response Messages

After you type a command, EXEC performs the command and may also acknowledge the command with either a message or an error message. EXEC error messages and error recovery are described in the manual *AOS/VS and AOS/VS II Error and Status Messages*.

By default, EXEC sends all messages to the system console (CON0 or PMAP0) — the terminal assigned to the master CLI or its son. If you have successfully issued an EXEC command from a different terminal, EXEC will acknowledge your commands on that terminal. Use the EXEC LOGGING command to change the terminal to which EXEC messages will be sent, and/or send copies of all status messages that EXEC generates to an EXEC log file. In any case, critical messages are always sent to the system console.

You can make EXEC messages brief or verbose, or you can silence EXEC messages, with the EXEC BRIEF, VERBOSE, or SILENCE commands. SILENCE is especially useful if your system console is a hardcopy terminal.

Often–used EXEC Commands

Table 3–2 shows the most popular EXEC commands, listed alphabetically.

Table 3–2 Often–Used EXEC Commands

Command	Explanation
ACCESS	Specifies user access to EXEC–controlled resources.
ALIGN	Pauses EXEC's printer–managing process (XLPT). Useful when someone wants to align the paper in the printer.
CANCEL	Cancels a request that is queued but not yet active. To stop an active request, use FLUSH.
CONTINUE	Continues (resumes processing in) a batch queue, specified stream, or device. Useful after EXEC startup (within UP.CLI macro) or after a queue, stream, or device has been paused.
DISABLE	Disables one or more user terminals for user logon. Useful when you plan to shut down EXEC, and don't want users logging on. Also useful when you want to free one or more EXEC–enabled terminals for use by a program, like DG/SNA or TPMS; or when you want to disable and then assign an unterminated line, so that no program can open that line.
ENABLE	Enables one or all user terminals for user logon via EXEC; primarily used within the UP.CLI macro.
FLUSH	Flushes (terminates) the job that a batch stream or device is processing. To cancel a request before it becomes active, use CANCEL.
FORMS	Allows you to specify a file to be used for special form printing; e.g., bills.
HALT	Terminates the EXEC process and all its sons, including all user processes. This command has no switches or arguments.

(continued)

Table 3–2 Often–Used EXEC Commands

Command	Explanation
MODIFY	Modifies queue entries for any inactive job. You can modify many parameters with the switches available for this command.
MOUNTED	Indicates that a tape has been mounted on a tape unit, in response to a user MOUNT request.
OPERATOR	Tells EXEC that an operator is on or off duty. Unless the operator has given ON notice to EXEC, users' MOUNT requests and batch jobs submitted with the /OPERATOR switch will not result in messages to the operator and, hence, will not be responded to. OPERATOR ON @CONn makes @CONn the current operator's terminal.
PAUSE	Pauses a batch queue, specified stream, or device in an orderly way, after the current request is finished. It prepares for normal EXEC shutdown or change in device specifications. Later, CONTINUE commands in UP.CLI or typed by OP or by a user with owner access to the paused resource will continue normal processing.
SILENCE	Tells EXEC to keep quiet; suppresses batch or queue messages to the system console and EXEC log file if active. Useful on hardcopy terminals.
START	Associates a queue with a device. The CONTINUE command then activates the device. Usually issued by the UP.CLI macro.
STATUS and SPOOLSTATUS	Displays information about batch queues and streams or queues and devices. The CLI command QDISPLAY/V is also handy for displaying information about queues and devices.

(concluded)

Managing User Logon

The following EXEC commands are used to manage user logon:

- ENABLE enables terminals for user logon
- CONSOLESTATUS displays terminal–user status
- TERMINATE terminates the user process associated with a terminal
- DISABLE disables a terminal; prevents user logon

The ENABLE and DISABLE commands are used in the UP.CLI and DOWN.CLI macros, first to enable all user terminals once the system is ready for general use, and then to prevent users from logging on so the system can be brought down in an orderly manner. These commands can also disable and enable specific terminals with the @CONn argument.

CONSOLESTATUS finds the username, PID, and CON number for any terminal — or for all terminals if none is specified. TERMINATE terminates the process running on a terminal. You can use it when a user program is hung and the user can't exit from a program normally; however, since it does not allow the program to make a normal exit, data can be lost and files impaired. Do not use TERMINATE in place of DISABLE.

Two other useful commands are SILENCE and UNSILENCE. When a console DISABLE command completes, a status message is sent to EXEC's logging console. You can suppress these messages using the SILENCE command with the argument @CONSOLE. To cancel the SILENCE command and have logoff messages once again sent to the logging console, use the UNSILENCE command — once again, with @CONSOLE.

Standard Logon Procedures

If you are not using a custom logon procedure (see the file LOGON_TOOLKIT.DOC in :UTIL:LOGON_TOOLKIT), a terminal that's turned on and enabled by EXEC displays the standard logon banner for the AOS/VS or AOS/VS II operating system that it is running; for example,

**** *AOS/VS II Rev n / Press NEW-LINE to begin logging on* ****

Or, a terminal may display a tailored logon screen instead (tailoring the logon screen is discussed in the *Installing* manual for your system).

To log on, a person presses NEW LINE on an enabled terminal; EXEC then prompts for a username and password. Users cannot log on under EXEC unless they have valid user profiles, created with PREDITOR. EXEC compares the username–password pair the user enters with the pair in the user's profile.

When the person types a valid username and password, EXEC tells the operating system to create a user process for that person, logs the user on, and sends the logon message to the user's terminal.

The operating system uses the parameters in the profile when it creates the user process, and it enforces these thereafter. Parameters in the profile include initial IPC filename, program to be executed for the user, number of sons, Superuser, Superprocess, and System Manager privileges, and the amount of disk space allowed, among other things. All these issues are covered in the previous chapter.

While the user process runs, EXEC records terminal use, pages printed, device time used for MOUNT requests, and privileged–user logons. It places this information in the system SYSLOG file if system logging is on. SYSLOG is different from EXEC's own log file.

When a user terminates his/her initial program (as by typing BYE to the CLI), EXEC terminates the user process. The user's screen displays

Process n terminated
Connect time H:MM:SS
(Other jobs, same USERNAME — Console: n, Batch: n)
User Username logged off @CONn DD–MON–YY HH:MM:SS

The terminal is then free for other users. Naturally, a privileged process like the master CLI or EXEC can terminate any user process at any time, but this should be done only if necessary.

Logon Errors

EXEC expects the user profile and user directory created by PREDITOR to be intact. If the user directory was somehow deleted, its ACL changed, or the profile tampered with, the user may receive an error message telling him or her to call the system manager.

Should this problem occur, run PREDITOR on the profile of this username. PREDITOR will then automatically rectify file problems. (But if the directory was inadvertently deleted, the user's files will need to be reloaded from backup media.)

Managing Queues and Devices

In the multiuser environment, users share the use of system resources like printers, plotters, batch processors, and operator assistance for mounting and dismounting tapes. EXEC coordinates users' requests — or *jobs* — by means of *queues* where the jobs wait to be processed. Each queue handles a specific type of job (print, batch, etc.) — although there can be more than one queue for a given job type.

You can have as many as 256 queues, and as many as 1024 jobs per queue.

EXEC refers to the printers, plotters, and other resources as *devices*. A batch processor is also considered a device.

Devices are controlled by cooperative programs which perform the processing appropriate for the device. XLPT, for example, is the cooperative which controls printers to perform print processing. All of the cooperative programs discussed in this chapter are listed in Table 3–3.

The remainder of this chapter covers the following topics:

- Configuring queue environments with EXEC
- Managing queue environments
- Special issues pertaining to printers, batch processors, tape management, and network processing
- A dictionary of EXEC commands

EXEC commands which apply to queues and devices in general are discussed first and summarized in Table 3–5; commands specific to certain types of processing are dealt with in the section on each type.

Configuring Queue Environments

Configuring your queue environment includes the following kinds of actions:

- Creating and opening queues (CREATE and OPEN commands);
- Specifying the devices with which the queues are to be associated (START command);
- Breaking the connections between queues and devices (STOP command);
- Closing and deleting queues (CLOSE and DELETE commands);
- Setting various parameters for devices and streams (commands listed in Table 3–4).

The following sections explain the use of these EXEC commands. For additional information and examples, refer to the "EXEC Commands" section.

Creating and Deleting Queues

When the operating system is brought up the first time, it automatically creates four queues: BATCH_INPUT, BATCH_LIST, BATCH_OUTPUT, and MOUNTQ. BATCH_INPUT is for user batch jobs, BATCH_LIST and BATCH_OUTPUT are print queues for list and output files for batch jobs, and MOUNTQ is for tape mount requests.

You will probably need to create at least one more queue — a print queue. To create queues, use the CREATE command. The format of the create command is as follows:

```
CX CREATE queue–type queuename
```

The queue–type should be one of the following: BATCH, PRINT, PLOT, MOUNT, FTA, SNA, HAMLET, and USER. Table 3–3 indicates the default CLI Q–series command, queue name, and cooperative process for each type of queue.

Table 3–3 Default Queue Names, Cooperative Processes and CLI Commands

Queue Type	Queued by CLI Command	Default Queue Name	Default Cooperative
BATCH	QBATCH	BATCH_INPUT	XBAT
PRINT	QPRINT	LPT	XLPT
PLOT	QPLOT	PLT	XLPT
MOUNT	MOUNT (@LMT)	MOUNTQ	XMNT
FTA	QFTA	FTQ	XNET
SNA	QSNA	SNQ	XNET
HAMLET	QSUBMIT	HAMQ	XNET
USER	User Defined	User Defined	User Defined

Queues are not deleted when the operating system is brought down; when the system comes up again they are still there with the jobs that were queued before the system was brought down.

To delete a queue, use the DELETE command. Its format is as follows:

```
CX DELETE queuename
```

The USER Queue Type

The USER queue type enables you to create a queue for jobs to be run by a special cooperative program tailored for your system, or not running under EXEC. To create a USER–type queue, simply specify USER as the queue type and assign whatever queue name you want. For information on starting a user cooperative on a USER queue, see the section "Starting Custom Cooperatives."

Opening and Closing Queues

A queue that has been newly created (including the default batch and mount queues) is closed — that is, jobs cannot be placed in the queue. To allow jobs to be placed in the queue, you must open it using the OPEN command. (However, queues are not closed automatically when the operating system is brought down; they will still be open when it comes back up.)

The format of the OPEN command is as follows:

```
CX OPEN queuename
```

To prevent jobs from being placed in a queue, or to prepare a queue to be deleted, use the CLOSE command. Its format is as follows:

```
CX CLOSE queuename
```

Note that if you intend to delete a queue, the queue must be be closed and empty. Use the CLOSE command to close the queue and use the PURGE command to empty it. The PURGE command has the following format:

```
PURGE queuename
```

Starting and Stopping Queues and Devices

When setting up a queue environment, you must identify which devices are associated with which queues. That is, you must specify which resources will service the requests that are waiting in EXEC's queues. This is done using the START command. START has the following format:

```
CX START queuename devicename
```

You can associate a new queue with a device that has already been started to another queue. For example, the default queues BATCH_OUTPUT and BATCH_LIST can be started on the line printer LPB (pathname @LPB); then, a print queue LPT can be created, opened, and also started on LPB. Jobs in all three queues will print on the same printer.

Also, you can associate a single queue to several devices. This is useful, for example, if you have two or more identical printers but want all your print requests to go through a single print queue.

The system automatically associates batch and mount queues with a "device" of the same name when it creates the queues. Thus, the BATCH_INPUT queue has a corresponding BATCH_INPUT device; if you create a new batch queue, for example one named BATCH2, it will be started on a BATCH2 device — you do not need to use the START command for batch or mount queues.

The device name given with START, STOP, and other EXEC commands that control devices, is in the form of a pathname when the cooperative controlling the device is XLPT or XNET. A printer device name usually consists of the "@" character (representing the :PER directory), followed by the device name (for example, @LPB or @CON27), but the pathname could also be a link. The device name for network processors FTA, SNA, and HASP II (HAMLET) identifies the IPC port of the network process; for example, @FTA_EXEC.

The device names for MOUNTQ, BATCH_INPUT, and any other batch queues are not in the form of pathnames; they are always the same as their queue names.

There are two optional switches for START that are not dependent on the type of queue being started. These are the /NAME and /STREAMS switches.

The /NAME switch specifies, by process name, the cooperative which should control the device being started. It is valid only when the device has not already been started. It is useful, for example, to balance the work required of cooperatives, or for putting a troublesome printer on a separate XLPT so that all printers do not have to be taken out of commission when it requires servicing.

If the /NAME switch is not used, and there is already a process of the same type running, EXEC assigns the new device to the same cooperative process. For example, you could start a second printer without the /NAME switch and have it run on the same XLPT process as the first printer, or start the second printer with the /NAME switch to have it run on a separate XLPT process.

The /STREAMS switch, also valid only when the device is not already started, is used to indicate how many jobs the device can process at once — that is, how many *streams* the device has.

The default number of streams for a new batch processor is one (but BATCH_INPUT is created with four streams); the default for mount and network processing is 10. Printers and plotters have only one stream each.

For more information on the device names to use with the START command, and the switches and arguments for modifying the command with printers, refer to the sections on the specific queue–types, and the START command description in the "EXEC Commands" section.

To reverse the effect of a START command, use the STOP command. The STOP command has the following format:

STOP *[queuename] [devicename]*

Either or both of the arguments can be specified. If only the queuename is given, the queue will no longer be associated with any devices. This must be the case before a queue can be deleted. If only the devicename is given, the device will no longer be associated with any queues.

If the effect of a STOP command leaves a device associated to no queues, the device becomes unknown to EXEC. That is, it can be used by other processes, can be assigned to a different cooperative, or, if it is a terminal, can be enabled for logon.

Starting Custom Cooperatives

Your site may have special job processing needs which are not offered by the default cooperatives XLPT, XBAT, XNET and XMNT. If this is the case, a custom cooperative program can use EXEC queues to manage its jobs. (See the on–line documentation :UTIL:COOP_TOOLKIT:COOP_TOOLKIT.DOC, which describes the EXEC Cooperative Toolkit you can use to develop such a cooperative program.)

To create a cooperative process using your custom program, first use CLI's PROCESS command. For example, if you have a custom print cooperative program :CUSTOM_COOPS:PRINT.PR, its process could be created as follows:

```
) PROCESS/NAME=CUSTOM_PRINT :CUSTOM_COOPS:PRINT.PR ↵
```

The START command to assign a printer to this cooperative uses the /NAME switch to identify the cooperative. Such a command might be:

```
) CX START/NAME=CUSTOM_PRINT LPT @LPB ↵
```

If the PROC command was issued from a process with a username other than EXEC's (usually OP), the START command should provide the full process name to the /NAME switch. For example, if a user PHIL issued the PROC command, the above START command would be altered as follows:

```
) CX START/NAME=PHIL:CUSTOM_PRINT LPT @LPB ↵
```

Starting Cooperatives Not in EXEC's Process Hierarchy

In another example, you could have a special high–speed printer, controlled by a dedicated XLPT process running under OP, instead of EXEC. To create a special queue named HIGH_SPEED, and then start printer LPB2 on this queue, type

```
) PROC/DEF/NAME=XLPT2 :UTIL:XLPT.PR ↵
) CX CREATE PRINT HIGH_SPEED ↵
) CX OPEN HIGH_SPEED ↵
) CX START/NAME=OP:XLPT2 HIGH_SPEED @LPB2 ↵
) CX CONTINUE @LPB2 ↵
```

Setting Device and Stream Parameters

After a device is started, you can set various parameters for each of the device's streams. The commands that set these parameters all have the following format:

CX command *[devicename] [stream–number] [additional–arguments...]*

The devicename and stream–number arguments are optional. If devicename is not used, BATCH_INPUT is the assumed device name. If stream–number is not used, all of the device's streams will be affected. The additional arguments depend on which command is being issued.

Table 3–4 describes stream parameters which are independent of the type of device associated with the stream. The table indicates their initial value when the device is started and which commands to use to change these parameters. Stream parameters that are specific to different device (and queue) types are described in the section on each type.

Table 3–4 Stream Parameters

Parameter/ Command	Initial Value	Description
PAUSE CONTINUE	PAUSE	Streams that are paused are prevented from running jobs.
LIMIT UNLIMIT	UNLIMIT	Prevents the stream from running jobs with limits greater than limit–value.
LIMIT	34:24:32 65535	Indicates the upper limit of CPU time or pages allowed a job if limiting is on.
SILENCE UNSILENCE	UNSILENCE	Silenced streams do not send messages to the logging console.
VERBOSE BRIEF	VERBOSE	Specifies the detail of messages sent to the logging console when silencing is off.
QPRIORITY	0–255	Restricts the stream to running jobs within the specified qpriority range.

Note that, since streams are initially paused, you must set the paused/continued parameter to "continued" before jobs will be allowed to run. This is done (normally, in the UP.CLI macro) using the CONTINUE command as follows:

CX CONTINUE *[devicename] [stream–number]*

Remember that if the devicename is not provided, BATCH_INPUT is assumed. Also, if the stream–number is not given, all streams of the device are affected.

Putting START and CONTINUE in Your UP.CLI Macro

When you have created and opened the queues needed on your system, you will want the necessary START and CONTINUE commands for these queues and their associated devices to be part of your UP.CLI macro. This way they will be given automatically each time your system comes up. The same is true for any nondefault device and stream parameters. Later on, if you add a device or a queue to your system, you will want to edit your UP.CLI macro, adding the necessary START and CONTINUE commands, and any other nondefault parameters.

For more information on creating the UP.CLI and DOWN.CLI macros, including sample macros, see the *Installing* manual for your system.

Managing a Queue Environment

Once you create your queues, devices, and streams and set parameters for them, EXEC's role consists mainly of ensuring the smooth processing of jobs once they enter their respective queues. The commands for managing the queue environment fall into three categories: obtaining status information, managing jobs, and managing devices.

Obtaining Status Information

EXEC and CLI provide several commands for determining the status of queues, devices, and cooperatives. Queue status is determined using CLI's QDISPLAY command. Details of this command can be found in the CLI manual or you can simply issue the command to see what it returns. Device status can be obtained by using EXEC's SPOOLSTATUS and STATUS commands.

The format of the SPOOLSTATUS command is as follows:

CX SPOOLSTATUS *[queuename or devicename]*

The queuename or devicename argument is optional. If it is not given, SPOOLSTATUS returns information on all devices known to EXEC. This information includes which queues are associated with a device or which devices are associated with a queue. Aside from indicating queue–device relationships, SPOOLSTATUS returns information about how a device is configured. Since this information depends of the type of device, this information is described in the appropriate section for the device type.

The format of the STATUS command is as follows:

CX STATUS *[devicename] [stream–number]*

The devicename argument is optional. If it is not provided, BATCH_INPUT is assumed. The stream–number argument is also optional. If it is not provided, STATUS returns information on all of the device's streams.

The information returned by STATUS includes whether the stream is paused or continued, whether it is limited and to what extent, and whether it is active. If the stream is active, STATUS returns various information that is specific to the device's type. Descriptions of this information are given in the appropriate type–specific section.

Managing Jobs

Using information acquired from a user or from the status calls described above, you may want to manipulate some of the jobs that are waiting in queues. EXEC's HOLD command prevents individual jobs from running without affecting other jobs in the queue. This format of the HOLD command is as follows:

CX HOLD *[sequence–number]*

This sets the job as "Held By Operator" as indicated by the "E" flag returned by CLI's QDISPLAY. Instead of using the sequence–number argument, you may use the /USERNAME switch, which holds all inactive jobs submitted by the specified user.

You can reverse the effect of a HOLD using the UNHOLD command. Its format is identical to HOLD, including use of the /USERNAME switch.

You can cancel individual inactive jobs at your discretion using the CANCEL command. Its format is also identical to the HOLD command. When upi cancel a job in this way, the job is marked "Cancelled by Operator" as indicated by the "F" flag returned by CLI's QDISPLAY.

In addition to holding, unholding, and cancelling jobs, you can modify an inactive job's parameters using the MODIFY command. The format of the MODIFY command is as follows:

CX MODIFY sequence–number

Switches on this command indicate which job parameter should be modified and what its new value should be. Parameters that can be modified include printing specifications, time of processing, and output file pathname. Please refer to the command dictionary section on MODIFY for a full list of these switches.

Managing Devices

Using information acquired from the status calls described above, from your own examination of your system's devices, or from a concerned user, you may want to clear a device or restart a job that is already running.

As described above, the CANCEL command cancels queued jobs that are waiting to be processed. The FLUSH command is used to clear active jobs. Instead of specifying the sequence–number required by the CANCEL command, FLUSH allows you to specify the device name and, optionally, a stream. The format of the FLUSH command is as follows:

CX FLUSH *[devicename] [stream–number]*

The stream–number argument is optional. If it is not provided, all streams of the device will be flushed. After the current active job is flushed, the device or stream will process the next job waiting in the queue.

Jobs in progress can be restarted using the RESTART command. Its format is as follows:

CX RESTART *[devicename] [stream–number] [begin–page] [end–page]*

Again, the stream–number argument is optional. If it is not provided, all jobs in progress on the device will be restarted. Use RESTART when you notice a problem after a job has begun (for example, a printer jams). RESTART accepts as additional arguments the page number(s) you want the job to begin and/or end with when it restarts. For example, if a printer runs out of paper halfway through a long print job, you don't need to start the whole job over after the paper is reloaded.

EXEC Commands—All Queues and Devices

Table 3–5 summarizes the commands that control cooperative processes in general. Later sections discuss EXEC commands specific to print, batch, networking, and mount jobs, in that order.

Table 3–5 EXEC Commands—All Queues and Devices

Type of Operation	EXEC Command	Brief Description
Control a queue	CREATE	Creates a queue.
	OPEN	Opens a queue to user requests.
	CLOSE	Closes a queue to user requests.
	DELETE	Deletes a queue.
Manage queue entries	MODIFY	Modifies a job.
	HOLD	Prevents a job from being processed.
	UNHOLD	Negates HOLD command.
	CANCEL	Cancels a job.
	PURGE	Deletes entries in a stopped queue.
Control link between queue and device	START	Associates a queue with a device.
	STOP	Dissociates queues and devices.
Control a device	PAUSE	Suspends processing at one or more streams.
	CONTINUE	Resumes a paused stream.
	RESTART	Restarts an active job.
	FLUSH	Flushes (kills) an active job.
Set device parameters	BRIEF	Makes status messages brief.
	VERBOSE	Makes status messages verbose.
	SILENCE	Suppresses device status messages.
	UNSILENCE	Negates SILENCE command.
	LIMIT	Enforces limits on CPU time (batch) or printed pages (print).
	UNLIMIT	Negates LIMIT command.
	QPRIORITY	Sets a new range of priorities that will be accepted by a stream.
Get status of queue or device	SPOOLSTATUS	Displays queue–to–device association and device status.
	STATUS	Displays status of one or more streams.
Control access to queue or device	ACCESS	Specifies user access to queues and devices.

The Queue Cleanup Program

EXEC stores all the temporary files used in managing queues in directory :QUEUE. (:QUEUE always includes the files QUEUES and JOBS, which manage queue information for EXEC.) The queue cleanup program, QCMP.PR, is a utility that scans :QUEUE and deletes unused files. QCMP can delete all files/directories that are not currently submitted to any queue or queued job — so don't let anyone use directory :QUEUE for file storage.

Usually, QCMP is executed via the UP.CLI macro before EXEC is brought up. You can also run QCMP while EXEC is running, but you should close all queues first to prevent inadverently deleting temporary files created while QCMP is running.

Execute QCMP by typing

XEQ QCMP*[/YES][/L[=pathname]]*

The /YES switch tells QCMP to delete unused files; otherwise QCMP asks for confirmation before it deletes them. /L[=pathname] sends output to the @LIST file or the file named in pathname if you include =pathname. If you omit /L, output goes to the system console.

QCMP scans the :QUEUE directory. Then if you omitted the /YES switch, it asks

May QCMP delete unused files in :QUEUE?

Type YES and press NEW LINE if you want to delete unused files; type NO (and press NEW LINE) if you do not. If you type YES or used the /YES switch, QCMP displays either

No unused files in :QUEUE to delete.

or

Deleted following unused file(s):

... (filenames deleted) ...

Note that QCMP deletes user files in :QUEUE.

If you type NO to the QCMP query, QCMP reports

Would have deleted the following unused file(s):

... (filenames) ...

When QCMP finishes, it returns you to the CLI.

Managing Print Processing

This section discusses using EXEC commands to manage print queues and devices.

Creating and Opening Print Queues

Usually, the primary print queue on a system — the one most frequently used — is named LPT. When a user types the QPRINT command without a /QUEUE switch, the print job is placed in queue LPT.

To create and open queue LPT you would type

```
) CX CREATE PRINT LPT ↵
) CX OPEN LPT ↵
```

If your system has additional printers, you create and open queues for them in the same way, assigning any queue names you want. For example, if your system has one or more letter–quality printers, you can create queues named LQP, LQP1, and so on for these. Users would queue print jobs to those printers by using the /QUEUE switch — for example

```
) QPRINT/QUEUE=LQP MYLETTER ↵
```

A queue cannot have the same name as a printer — because both names are files in :PER.

Starting and Continuing Print Devices

The START command associates a print queue with a particular print device and an XLPT cooperative process. If an XLPT process is not running and you do not specify a process name, EXEC will create one with the process name OP:XLPT1. If you want to specify the process for this device, use the /NAME switch with the START command. A single XLPT process can manage up to 15 print devices simultaneously.

For a device to process jobs received from a queue, the device must be continued with EXEC's CONTINUE command.

For example, to start printer LPB on queue LPT and continue the printer, you would type

```
) CX START LPT @LPB ↵
) CX CONTINUE @LPB ↵
```

To start a letter–quality printer connected to console line 23 (CON25) on queue LQP, type

```
) CX START LQP @CON25 ↵
) CX CONTINUE @CON25 ↵
```

You can start two or more printers on a single queue; for example

```
) CX CREATE PRINT EITHER ↵
) CX OPEN EITHER ↵
) CX START EITHER @LPB ↵
) CX START EITHER @LPB1 ↵
) CX CONTINUE @LPB ↵
) CX CONTINUE @LPB1 ↵
```

This command sequence creates queue EITHER, which sends output to either printer. Users can access this queue via the QPRINT/QUEUE=EITHER command/switch combination.

Instead of a printer, you can specify a disk file in the START command. Just make sure the file exists in directory :PER; then specify the filename instead of a printer name. For example, to link queue LPT to disk file LPTFILE, so that jobs queued to LPT are sent to file LPTFILE instead of a printer, type

```
) CX START LPT @LPTFILE ↵
) CX CONTINUE @LPTFILE ↵
```

Setting Special Printing Parameters

There are a number of special printing parameters that you can set, either with the START command or with other EXEC commands.

Switches used with the START command to set printing parameters are as follows:

/7BIT — Tells the printer to use only the rightmost 7 bits when it prints characters. Seven bit printing is the default XLPT uses.

/8BIT — Tells XLPT to use all 8 bits when printing. Use this switch if you have an 8–bit printer and want it to print 8–bit characters exclusively. If users will specify 8–bit printing only for some jobs on this printer, they can use the /8BIT switch with the CLI QPRINT command.

/NL — Tells XLPT to convert all NEW LINE characters to Carriage Return/NEW LINE sequences. Use this switch on any printer that does not automatically position at the start of a line after printing a NEW LINE character.

To select a printing forms file — such as a mapper file, which specifies character conversions — give the forms filename as an argument with the START command. The forms file must exist in directory :FORMS. For example, if you have an uppercase–only line printer, type the following to use the mapper file UPPER, which converts all lowercase characters to uppercase.

```
) CX START LPT @LPB UPPER ↵
```

Another way to specify a mapper file is with EXEC's MAPPER command (see Table 3–6). For more information on creating and using forms files, refer to the section on FCU (Forms Control Utility) in *Using the CLI (AOS/VS and AOS/VS II).*

The EXEC commands specific to printing are briefly described in Table 3–6. For details, see the description of each command in the "EXEC Commands" section.

Table 3–6 EXEC Commands for Print Processing

EXEC Command	Brief Description
ALIGN	Stops or continues the line printer (used to align it).
BINARY	Allows a printer to output a /BINARY or /PASSTHRU print job as well as normal print jobs.
CPL	Specifies the maximum number of printed characters per line.
DEFAULTFORMS	Specifies a forms files to be used by default.
ELONGATE	Turns elongated printing on or off for LP2/TP2 printers.
EVEN	Turns even pagination on or off for a printer.
FORMS	Specifies a forms file to be used for printer format control.
HEADERS	Changes the number of printed header pages.
LPP	Changes the maximum number of printed lines per page.
MAPPER	Specifies a mapper file for a printer.
TRAILERS	Changes number of printed trailer pages.

All of these commands take a print device name — for example, @LPB — as an argument. Some take other arguments and switches as well. Refer to the "EXEC Commands" section for details.

Managing Batch Processing

When someone wants to run a program, he or she can run it interactively from a terminal or in batch mode. Any user with the Use Batch privilege can use batch, by typing the CLI QBATCH command, followed by the desired CLI command. (QBATCH is described in the manual *Using the CLI (AOS/VS and AOS/VS II)*).

Default Batch Queues

When you first install and bring up the operating system, it creates four queues:

- BATCH_INPUT
- BATCH_LIST
- BATCH_OUTPUT
- MOUNTQ

BATCH_INPUT is the default queue to which batch jobs queued by users are sent for batch processing; it is created with four streams and started automatically (its device name is also BATCH_INPUT). BATCH_LIST and BATCH_OUTPUT are the print queues for all batch list and output files (unless the user specifies a different list or output file, using a switch with the QBATCH command). MOUNTQ is discussed later.

When the operating system is brought up the first time, you would open the batch queues and start BATCH_LIST and BATCH_OUTPUT on a printer. For example:

```
) CX OPEN BATCH_INPUT ↵
) CX CONTINUE 1 ↵

) CX OPEN BATCH_LIST ↵
) CX START BATCH_LIST @LPB ↵

) CX OPEN BATCH_OUTPUT ↵
) CX START BATCH_OUTPUT @LPB ↵
```

Note that, since BATCH_INPUT is the default queue for the CONTINUE command, you don not need to specify it. Also, in this example only stream 1 is continued. BATCH_INPUT is created with four streams (that is, it can potentially process four batch jobs simultaneously); if no stream is specified with the CONTINUE command, all four are continued.

In this example, BATCH_LIST and BATCH_OUTPUT are both started on printer LPB; other queues, such as the default print queue, may also be started on this printer. Once the printer is continued it can process jobs from all of its queues.

Once your batch queues are opened, they will stay open unless you close them — even when the system is brought down and up again. However, you will need to issue the CONTINUE command for BATCH_INPUT, and the START and CONTINUE commands for BATCH_OUTPUT and BATCH_LIST, each time the system comes up. Normally, this is done by incorporating these commands in your UP.CLI macro.

Creating Additional Batch Input Queues and Streams

If your system frequently uses batch processing, you might want to create additional batch queues. The following example creates and opens a batch queue named B1; its corresponding device, also called B1, has two streams. Both streams are then continued (if no stream is specified with the CONTINUE command, all streams are continued).

```
) CX CREATE/STREAMS=2 BATCH B1 ↵
) CX OPEN B1 ↵
) CX CONTINUE B1 ↵
```

Users can place batch jobs in the new queue by using the switch /QUEUE=B1.

More information on the CREATE, OPEN, START, and CONTINUE commands is found in the earlier section "Managing Queues and Devices," and also under each command name in the section "EXEC Commands."

The EXEC commands specific to batch processing are briefly described in Table 3–7. For details, see the description of each command in the "EXEC Commands" section.

Table 3–7 EXEC Commands for Batch Processing

EXEC Command	Brief Description
BATCH_OUTPUT	Specifies a print queue for batch output files from a specified batch input queue. If this command is not used, batch output files will go to the default BATCH_OUTPUT queue, or else to the queue specified by the user with the /QOUT= switch;
BATCH_LIST	Specifies a print queue for batch list files from a specified batch input queue. If this command is not used, batch list files will go to the default BATCH_LIST queue, or else to the queue specified by the user with the /QLIST= switch;
PRIORITY	Sets the process type (pre–emptible, resident, or swappable) and system priority for the process at the stream specified (if there is only one active batch stream, users need not specify the stream).

Managing Communications and Network Queue Processing

Data General's IBM emulators — DG/SNA and HASP II (HAMLET) — and the XODIAC networking agent FTA need EXEC queues to run properly. The queue names for each network processor are as follows:

Queue Type	Queue Name
HAMLET	HAMQ
SNA	SNQ
FTA	FTQ

To find the device name to use for your network processor, refer to the documentation provided with it.

The following commands prepare a networking processing environment which includes an FTA queue named FTQ and an FTA device called @FTA_EXEC, which can process up to 10 jobs at a time (the default number of network processing streams is 10).

```
) CX CREATE FTA FTQ ↵
) CX OPEN FTQ ↵
) CX START FTQ @FTA_EXEC ↵
) CX CONTINUE @FTA_EXEC ↵
```

Once you issue these commands, users can place FTA jobs in the new queue using the CLI command QFTA/QUEUE=FTQ.

If you want your network processor queue to be activated each time you bring up your operating system, you can insert the START and CONTINUE commands in your UP.CLI macro.

There are no EXEC commands which have special meanings for FTA, SNA, or HAMLET queues, devices, or streams. For more information on CREATE, OPEN, START, CONTINUE, and other commands which you can use with network processing, such as SILENCE, LIMIT, FLUSH, and PAUSE, see the section "Managing Print Queues and Devices," earlier in this chapter.

Managing Mount Processing

Mount processing allows users to request an *operator* to mount tapes on tape units, so that they can dump or load files for backup or other purposes. When they have finished, the users request that the operator dismount the tapes. The users make these requests with the CLI MOUNT and DISMOUNT commands (or, with another CLI command, specifying a pathname that begins with @LMT). For information on MOUNT and DISMOUNT, refer to the manual *Using the CLI (AOS/VS and AOS/VS II)*.

EXEC's mount environment includes a mount queue named MOUNTQ, a mount device, also named MOUNTQ, and a person designated as mount operator. The cooperative program for mount processing is XMNT.PR. The operator (assumed for this explanation to be you, the system manager), uses EXEC commands to handle mount jobs.

To coordinate mount processing, EXEC and XMNT.PR receive CLI commands from users and EXEC commands from the operator, and send messages to users and the operator. Details are given below.

MOUNTQ is created and started automatically when the operating system comes up the first time; you need to open it and give the EXEC OPERATOR ON command, as follows. (You do not need to give the CONTINUE command for MOUNTQ; the queue is continued automatically when you give the OPERATOR ON command.)

```
) CX OPEN MOUNTQ ↵
) CX OPERATOR ON @CON0 ↵
```

Once opened, MOUNTQ does not normally need to be recreated, reopened, or restarted when the system is brought down and up again. The OPERATOR ON command should be included in your UP.CLI macro.

If the XMNT process aborts, for example, you could shut the system down and restart it. But it would be less disruptive to simply restart the MOUNTQ by typing the following commands:

```
) CX START MOUNTQ MOUNTQ ↵
) CX OPERATOR ON @CON0 ↵
```

Details of the OPEN command are given in the earlier section "Managing Queues and Devices." Also see that section for a general discussion of managing and trouble–shooting jobs, queues, devices, streams, and cooperatives.

The remainder of this section discusses how to perform the duties of a mount operator. Table 3–8 lists the commands which apply specifically to mount processing.

Table 3–8 EXEC Commands for Mount Processing

EXEC Command	Brief Description
ALLOCATE	Restores a tape unit to the list of mountable units
DISMOUNTED	Tells EXEC that a tape has been physically removed from a tape drive.
MOUNTED	Tells EXEC that a tape has been physically mounted on a tape drive.
MOUNTSTATUS	Displays MOUNT request status.
OPERATOR	Tells EXEC that an operator is on or off duty.
PREMOUNT	Tells EXEC that a person has physically mounted a tape before a MOUNT request occurred.
REFUSED	Tells EXEC and a user that you refused the user's MOUNT request.
RELEASE	Removes a tape unit from the list of mountable units.
UNITSTATUS	Describes the mount status of tape units.

Viewing and Changing the List of Mountable Units

When the cooperative program XMNT.PR is started by EXEC (done automatically when the system comes up), it scans the directory :PER and makes a list of all MTU type files. This list represents the pool of mountable tape units potentially available for use by the mount operator. ("Potentially" because it is the list of units supportable by your tape controller(s) — not just the units your system actually has.) Use the UNITSTATUS command to display the list. The format of the UNITSTATUS command is as follows:

CX UNITSTATUS *[unit–name]*

The unit–name is optional. If it is not specified, the status of all units in the list is returned. The following is an example of output returned from such a UNITSTATUS command:

@MTB0 Not Mounted
@MTB1 Not Mounted
@MTD0 Mounted MID=45 USER=SCOTT PID=28

This system has two MTB tape units and one MTD tape unit. Units MTB0 and MTB1 have no tapes mounted, and unit MTD0 has an unlabeled tape mounted. (Note that an AOS/VS system with MTB and MTD tape controllers would show entries for MTB0 through MTB7 and MTD0 through MTD3, even though the site might have fewer units of each type.)

You can add a unit to the list with the ALLOCATE command, or remove a unit from the list with the RELEASE command. Using RELEASE, you could remove a faulty tape unit from the unit list to avoid mounting errors. Once the unit has been repaired, you could use ALLOCATE to again make it eligible for tape mounts. (Under AOS/VS, you could also release all units which do not represent actual tape units on the system, so that they do not show up on the UNITSTATUS list.)

Releasing an existing tape unit does not affect its availability for other purposes, such as DUMP and LOAD — only its availability to XMNT.

Managing Tape Mount Requests

A major duty of a system operator is to mount and dismount tapes as required to perform file backup and recovery. Users' CLI MOUNT and DISMOUNT commands are passed to the operator as mount and dismount requests from XMNT. The requests appear as messages on the system console, or else on the console that the operator has specified with the OPERATOR ON command.

A mount request is a message to the mount operator from XMNT to mount a tape on an available ("Not Mounted") unit. There are three types of messages that XMNT can send to the mount operator when making a mount request. The names of these message types are Unlabeled Mount Request, Labeled Mount Request, and Next Volume Request. There is only one type of dismount request, which is a Unit Dismount Request.

Unlabeled Mount Requests

An Unlabeled Mount Request is simply a request to mount an unlabeled tape on an available unit. For example, if user Jonathan had typed

) MOUNT MYTAPE ⤵

A mount request like this would appear on the system console or the console where someone had typed CX OPERATOR ON @[!CONSOLE]:

From Pid 0005 : (XMNT) 22–Nov–1993 13:30:00
****
Unlabeled Mount Request
****
MID = 3442 USER = JONATHAN
User Pid = 49 Requestor Pid = 55
From Pid 0005 :
Settings: Default Density

Respond: CX MOUNTED @unitname
or CX REFUSED

The following fields provide specific information about this mount request:

MID = is the sequence number of the mount job for which this request was generated. (This sequence number is also displayed in response to the MOUNTSTATUS command or the CLI QDISPLAY command.)

USER = is the username of the process which queued the job.

User Pid = is either the requestor pid, or (as in the example) it is the pid of the immediate son of EXEC which is the ancestor of the requestor pid.

Requestor Pid = is the pid of the process that issued the MOUNT command (or referenced @LMT).

Since the user did not specify a tape label or indicate, by including a message with the MOUNT command, that a particular tape should be mounted, EXEC displays the banner *Unlabeled Mount Request* so that the operator knows that a scratch tape is wanted. The mount operator should get a scratch tape, mount the tape on an available tape unit, and type the CX MOUNTED command, giving as an argument the unit name; for example,

) CX MOUNTED @MTB0 ⤵

XMNT will then send a message to the user that the tape is mounted and ready. The MOUNTED command will cause XMNT to dedicate the unit to the user who queued the mount job. To do this, XMNT temporarily changes the unit ACL such that only the user has access.

If the mount operator is not going to mount the tape, he/she responds CX REFUSED. The REFUSED command tells XMNT not to make any further mount requests for this mount job, and to send a message to the requestor indicating that the request was refused.

Labeled Mount Requests

A Labeled Mount Request is a request to mount a labeled tape on an available unit. For example, if user Jonathan had typed

) MOUNT/VOLID=VOL0/VOLID=VOL1 MYTAPE PLEASE USE LARGE TAPES. ↵

A mount request like this would appear on the system console (or the console where someone had typed CX OPERATOR ON @[!CONSOLE]):

From Pid 0005 : (XMNT) 22–Nov–1993 13:30:00
****
Labeled Mount Request
****
MID = 3448 USER = JONATHAN
User Pid = 49 Requestor Pid = 55
Volumes: VOL0, VOL1

From Pid 0005 :
Mount Volume: VOL0
Settings: Default Density
User Message: PLEASE USE LARGE TAPES.

Respond: CX MOUNTED @unitname
or CX REFUSED

Fields specific to a labeled mount request are as follows:

Volumes: displays a list of labels for tapes that this mount job can use.

Mount Volume: displays the label for the tape that the mount operator should mount at this time.

For this request, the mount operator should find the tape with the label VOL0 and mount it on a free unit. If there is no prelabeled tape, the operator is expected to label a scratch tape with the label VOL0 using the LABEL utility (LABEL.PR). For example,

) X LABEL @MTB0 VOL0 ↵

would write the label VOL0 on a scratch tape on unit MTB0. The operator should then respond using the MOUNTED command. Again, the operator may choose to refuse the request using the REFUSED command.

Next Volume Requests

A Next Volume Request is a request to mount the next volume of a labeled tape on an available unit. A typical next volume request would look like this:

```
From Pid 0005 : (XMNT) 22-Nov-1993 13:30:00
                       **********
                       Next Volume Request
                       **********
                       MID =    3453     USER = JONATHAN
                       User Pid =   49     Requestor Pid =   37
                       Volumes:      VOL0, VOL1
                       Units:        @MTB0 / VOL0
From Pid 0005 :
                       Mount Volume:  VOL1
                       Settings:      Default Density
                       User Message:  PLEASE USE LARGE TAPES.
                       Default Unit:  @MTB0

                       Respond:       CX MOUNTED [@unitname]
                                   or CX REFUSED
```

The *Units:* field displays a list of units that are currently mounted for this job, and the labels of the tapes that are mounted on each.

For this request, the mount operator prepares a tape with the label VOL1. To mount the tape, he/she can remove a tape from one of the units in the mount job's list of units and place the tape there, or place the tape on an unmounted unit. In either case the mount operator will then respond with the MOUNTED command. Again the operator may choose to refuse the request with the REFUSED command — although to do so at this point when the user is making a backup would also make the previous volume(s) unusable.

If the user specified all volumes needed, the XMNT process keeps track of the volumes and sends each Next Volume Request automatically; the user does not need to type any more commands until all volumes are completed and he or she gives the CLI DISMOUNT command.

Additional Mount Request Messages

The previous mount request examples did not show all of the messages that can appear in a mount request. Some of the additional fields can occur on any type of mount request and one can only occur on the labeled tape mount requests (labeled Mount Request, Next Volume Request).

The following example is a labeled Mount Request which displays all of the fields that can appear in any mount request.

```
From Pid 0005 : (XMNT)  22-Nov-1993  13:30:00
                        **********************
                        Labeled Mount Request
                        **********************
                        MID =     3448      USER = JONATHAN
                        User Pid =   49     Requestor Pid =   55
                        Volumes:       VOL0, VOL1
From Pid 0005 :
                        Mount Volume:  VOL0
                        Settings:      1600 BPI, IBM Format, Write-Protect
                        User Message:  Thanks.

                        Respond:       CX MOUNTED @unitname
                                    or CX REFUSED
```

The fields are as follows:

n BPI	tells the mount operator the density setting for the unit that the user would like. The possible values for this field are 800, 1600, 2400, and ADM (automatic density matching).
IBM Format	informs the mount operator that the user would like to use a unit capable of processing IBM Format tapes.
Write Protect	informs the mount operator that the user has only requested read/execute (RE) access to the mount device (by default the user obtains WARE access to mounted units).
User Message:	displays a message the user had supplied when submitting the mount job.

The next example is a Next Volume Request which illustrates a type of request called an "extending mount request."

```
From Pid 0005 : (XMNT)  22-Nov-1993  13:30:00
                        **********
                        Next Volume Request
                        **********
                        MID =     3453      USER = JONATHAN
                        User Pid =   49     Requestor Pid =   37
                        Volumes:      VOL0, VOL1, (May Extend)
                        Units:        @MTB0/VOL1
From Pid 0005 :
                        Mount Volume:  Any Volume
                        Settings:      Default Density

                        Respond:       CX MOUNTED @unitname volumename
                                    or CX REFUSED
```

In this example the *Mount Volume:* prompt indicates that the operator should mount any volume. This means that the user has used the /EXTEND switch. An extending mount request is only possible when the user has used the /EXTEND switch with the CLI MOUNT command. Otherwise, if the last volume specified is used up before all the files are dumped, the whole process may have to be started over because it will not be able to complete normally.

Now it is up to the operator to label a tape with an appropriate label, mount it, and respond by issuing the CX MOUNTED command, specifying a unit name and the name of the volume just labeled.

In the event that the XMNT process receives an error from a program that writes to a tape, it will display an *Error:* message and then redisplays the same mount request for this mount job to which the mount operator last responded. This gives the operator an opportunity to retry mounting the tape. An *Error Occurred* message can only appear on a labeled mount request.

Managing Dismount Requests

A dismount request is a message from the XMNT process to the mount operator to remove the tape(s) from the specified unit(s). This is an example of a typical dismount request.

```
From Pid 0005 : (XMNT)  22-Nov-1993  13:30:00
                        **********************
                        Unit Dismount Request
                        **********************
                        MID =     3490      USER = JONATHAN
                        User Pid =    49      Requestor Pid =    49
                        Volumes:        VOL0
                        Units:          @MTB0/VOL0
From Pid 0005 :
                        Dismount Unit:  @MTB0
                        User Message:   Please mark the tape as mine.

                        Respond:        CX DISMOUNTED [@unitname]
```

This example includes the fields *Dismount Unit:* and *User Message*.

Dismount Unit: displays a list of tape units that are ready to be dismounted.

User Message: only appears on the dismount if the user has specified that a message string should be passed to the mount operator.

The operator should remove the tapes from these units. The mount operator can respond CX DISMOUNTED @unitname for each unit he/she has dismounted, or can respond CX DISMOUNTED with no argument to tell the XMNT process that all of the dismountable units have been dismounted for this mount job. After receiving dismounted responses from the mount operator, the XMNT process resets the ACLs on the tape unit or units. After all of a mount job's units have been dismounted, the mount job is complete, and it is removed from the MOUNTQ.

Terminated User Dismount Request

The XMNT process checks all of its user pids whenever it receives a new mount job, a MOUNTSTATUS command, or a UNITSTATUS command. When the XMNT process discovers that the owner of a mount job's user pid has terminated, it automatically generates a dismount request for the user's mount job.

```
From Pid 0005 : (XMNT)  22-Nov-1993  13:30:00
                        **********************
                        Unit Dismount Request
                        **********************
                        MID =     3491      USER = JONATHAN
                        User Pid = (User Terminated)
                        Volumes:        VOL0
                        Units:          @MTB0/VOL0
                        Dismount Unit:  @MTB0

                        Respond:        CX DISMOUNTED [@unitname]
```

Dismount requests can be for either labeled or unlabeled tape mount jobs.

Managing Inactive Requests

The XMNT process will display only one request at a time. This request is called the active request. The active request is displayed repeatedly, with longer delays between repetitions, until the mount operator responds to the request. While the XMNT process is waiting for a response on the active request, it can be internally queueing up other requests for the mount operator. When the XMNT process receives a response on the active request, the next queued request then becomes the active request and is displayed.

The operator may want to deal with requests other than the active request. The MOUNTSTATUS command gives the current state of every tape mount job, including the MID (the job's sequence number). When responding to mount requests other than the active request, the operator can use the /MID= switch to refer to the mount job. For example:

```
) CX MOUNTED/MID=48 @MTB0 ↵
```

If you are giving a DISMOUNTED command, the XMNT process does not need to know the MID when you specify a unitname. But, if you want to tell the XMNT process that all units for the mount job have been dismounted, you can use the MID as an argument to the command.

The XMNT process handles responses to inactive requests the same as responses to active ones. The current active request will continue to display on the operator's console.

Removing Queued Mount Jobs

Ordinarily, a job is automatically removed from the MOUNTQ when a CX REFUSED or a CX DISMOUNTED is received by the XMNT process. If the mount operator wants to remove an inactive job from the MOUNTQ, or for some other reason REFUSED and DISMOUNTED can not be used successfully, the mount operator can use the FLUSH or CANCEL command. Refer to the sections "Managing Queues and Devices" and "EXEC Commands" for details on these commands.

EXEC Commands

The rest of this chapter describes all the EXEC commands in alphabetical order. With the exception of XHELP, each EXEC command begins with CONTROL @EXEC or its abbreviation, CX. You can abbreviate any EXEC command to its shortest unique string.

ACCESS

Specifies user access to EXEC–controlled queues and devices.

Format

CX ACCESS *[/D][/K]* { queuename / devicename } *[username,ACL]*

where

/D tells EXEC to set the ACLs to the default for the specified queue or device. The default ACL depends on the type of argument supplied, as described below.

/K tells EXEC to delete the ACL for the specified queue or device. The /K switch denies access to everyone but a user with the same name as EXEC, usually OP.

queuename is the name of the queue for which you're specifying access.

devicename is the name of the device for which you're specifying access.

username,ACL is a username/ACL pair or list of pairs specifying the access rights of each person listed. If omitted, EXEC displays the current access list for the queue or device.

Description

The ACCESS command lets you specify access to queues and devices by indicating nondefault ACLs for them. In this way, you can either limit or extend access to specified users.

To place entries into a queue, a user needs write (W) access to that queue. To use the CLI QDISPLAY command, a user needs read (R) access to the queue(s) requested. By default, EXEC sets the ACL for each queue to +,WR. To issue CONTROL @EXEC commands to a queue or device, a user needs owner (O) access to the queue or must be logged on as OP.

For printers and other cooperative devices, the ACLs of the associated queue(s) determine the ability to submit print or batch jobs (with CLI commands such as QPRINT, QPLOT, and QBATCH). To issue EXEC commands to a printer, a user needs owner access to the device.

If you give a user read access to a device, he or she can use the SPOOLSTATUS and STATUS commands to check the status of the device.

EXEC sets the ACL of the applicable :PER entry to the ACL you specify. The PREDITOR profile settings are unaffected by resource access granted by this command.

Access rights for queues are reset to the default each time you bring up the operating system. Therefore, edit your system's UP.CLI macro to include any CX ACCESS command lines that you want to retain permanently.

Only someone with the same username as EXEC, usually user OP, can issue EXEC commands that are not directed at a specific queue or printer. OP is also the only user who can issue EXEC commands to control terminals or magnetic tape units.

If you want to run a secure system, make sure that the UP.CLI macro eliminates user access to the MOUNTQ before UP.CLI enables terminals. UP.CLI can do this through the command ACL @MOUNTQ OP,WR. Then, only user OP or a superuser will be able to post mount requests via the MOUNT command. You can also restrict access to EXEC commands by using the 32-bit CLI LOCK/CX command, described in the manual *Using the CLI (AOS/VS and AOS/VS II)*.

The EXEC commands that other users cannot issue are

ALLOCATE	ENABLE	MOUNTED	REFUSED
CONSOLESTATUS	HALT	MOUNTSTATUS	RELEASE
CREATE	LOGGING	OPERATOR	TERMINATE
DISABLE	MDUMP	PREMOUNT	UNITSTATUS
DISMOUNTED	MESSAGE	PROMPTS	

Table 3-9 shows the EXEC commands that selected users can issue, provided that you give them owner access to a specified queue or printer.

Table 3-9 EXEC Commands Users Can Issue

EXEC Command	Queue or Device
ACCESS	queue or device
ALIGN	printer
BATCH_LIST	batch and print queues
BATCH_OUTPUT	batch and print queues
BINARY	printer
BRIEF	batch queue or device
CANCEL	queue containing job
CLOSE	queue
CONTINUE	batch queue or device
CPL	print device

(continued)

Table 3–9 EXEC Commands Users Can Issue

EXEC Command	Queue or Device
DEFAULTFORMS	print device
DELETE	queue
ELONGATE	print device
EVEN	print device
FLUSH	device
FORMS	print device
HEADERS	print device
HOLD	queue containing job
LIMIT	device
LPP	print device
MAPPER	print device
MODIFY	queue containing job
OPEN	queue
PAUSE	device
PURGE	queue
PRIORITY	device
QPRIORITY	device
RESTART	queue or device
SILENCE	device
SPOOLSTATUS	queue or device(s)
START	queue and device
STATUS	device
STOP	queue and/or device
TRAILERS	print device

(continued)

ACCESS (Continued)

Table 3–9 EXEC Commands Users Can Issue

EXEC Command	Queue or Device
UNHOLD	queue containing job
UNLIMIT	batch queue or print device
UNSILENCE	batch queue or device
VERBOSE	batch queue or device

(continued)

Why Use It?

Use the ACCESS command to establish user access to and control of queues and printers. You might want to use the ACCESS command to give several people the ability to use EXEC commands for your system's devices. Giving users the ability to use EXEC commands enables them to try to solve device problems in your absence. In order to issue EXEC commands the specified users need owner (O) access to the device or to the associated queue(s).

To see the current access control list for a queue or printer, omit the username argument.

Example

To see the current access control list for a queue named BATCH2, you'd type

```
) CX ACCESS BATCH2 ↵
```

SYSMGR,OWR +,WR

You could give users Jake and Kim the ability to control the BATCH2 queue by typing

```
) CX ACCESS BATCH2 JAKE,O KIM,OWR +,WR ↵
```

The command below lets all users issue CONTROL @EXEC commands to control the printer.

```
) CX ACCESS @LPB +,O ↵
```

ALIGN

Halts or continues the printing process so you can align paper.

Format

CX ALIGN *[/CONTINUE]* @printername *[[–]n]*

where

/CONTINUE continues the device.

@printername is the pathname of a print device in the :PER directory; for example, @LPB.

n is the page number where you want XLPT to restart processing. A positive *n* indicates an absolute page number in the file. A negative *n*, like –1, indicates a page *before* the last page printed.

Description

The ALIGN command tells XLPT to stop printing. For most print jobs, printing stops at the end of the currently–printing page. For jobs queued with /BINARY or /PASSTHRU, printing stops at the end of the current job, or, if multiple copies were queued, the end of the currently printing copy. To check the page XLPT was printing when the printer stopped, use the EXEC command STATUS devicename.

When you reissue ALIGN, you can specify where XLPT should restart the job. To restart, use the format

CX ALIGN/CONTINUE @printername *[[–] n]*

Why Use It?

Printer paper may be out of alignment, or the paper may jam. The ALIGN command stops the printer so you can align paper or do other things with the printer and then resume printing.

Example

If you are printing a large file on a printer named @LPB, type

) CX ALIGN @LPB ↵

When you have corrected the jam, continue printing by typing

```
) CX ALIGN/CONTINUE @LPB ↵
```

This command restarts the active job at the page where it stopped. If printing stopped on page 10, however, you might want to restart printing on page 9, instead of on page 11. In that case, you'd type

```
) CX ALIGN/CONTINUE @LPB -1 ↵
```

XLPT would then restart printing at the page where it stopped, minus one. Specifying -1 is the same as specifying 10–1, or page 9.

ALLOCATE

Restores a tape unit to EXEC's list of mountable units (opposite of RELEASE).

Format

CX ALLOCATE @unitname

where

@unitname is the name of a tape unit, preceded by @; for example, @MTB1.

Description

The ALLOCATE command tells the XMNT.PR to restore a tape unit to its list of mountable units. By default, all tape units for each tape controller are included on EXEC's list of mountable units. Thus this command is needed only if a unit has been removed from the list with the RELEASE command.

Why Use It?

You use ALLOCATE to restore a tape unit to the list of mountable units if you previously removed the unit by using the RELEASE command. When you bring XMNT down and then up again, all files of type MTU in :PER are placed into XMNT's list of mountable units.

Example

) CX UNITSTATUS ↵	UNITSTATUS displays all mountable units.
@MTB0 Not Mounted	It displays unit 0 only.
) CX ALLOCATE @MTB(1,2,3,4,5,6,7) ↵	Restore all units to mountable list.
) CX UNITSTATUS ↵	Try UNITSTATUS again.
@MTB0 Not Mounted *@MTB1 Not Mounted* ... *@MTB7 Not Mounted*	It displays all units.

BATCH_LIST

Sends all list files from a batch input queue to the specified print queue.

Format

CX BATCH_LIST { *[batch–queue]* print–queue
/DEFAULT *[batch–queue]* }

where

batch–queue is the name of the batch input queue whose list files you are redirecting. If omitted, the default batch queue is BATCH_INPUT.

print–queue is the name of the print queue you're associating with the specified batch queue or with BATCH_INPUT.

/DEFAULT tells the XBAT process to use BATCH_LIST as the print queue.

Description

This command changes the print queue for listings for the specified batch queue. The print queue you specify must exist as a queue of type PRINT.

With the /DEFAULT switch, this command restores the print queue to its original default, BATCH_LIST.

To use any batch input queue other than the default (BATCH_INPUT), a user must use the QBATCH command with the /QUEUE=queuename switch. Also, a user can specify a list file instead of the default print queue with the QBATCH command /QLIST=pathname switch. The file specified may not be a queue.

Why Use It?

The BATCH_LIST command gives you greater control over placement of batch listings. You can use this command to send all batch listings to a specific queue.

Example

The following commands create a new batch input queue and new batch list and output print queues, direct batch listings and output to the new print queues, and start the output and list print queues, as well as print queue LPT1, on the second line printer (LPB1).

) CX CREATE BATCH BATCH_IN2 ↵	Create batch input queue.
) CX OPEN BATCH_IN2 ↵	Open the queue for input.
) CX CREATE PRINT BATCH_LIST2 ↵	Create batch list print queue.
) CX CREATE PRINT BATCH_OUT2 ↵	Create batch output print queue.
) CX BATCH_LIST BATCH_IN2 BATCH_LIST2 ↵	
) CX BATCH_OUTPUT BATCH_IN2 BATCH_OUT2 ↵	Change list and output print queue names for new input queue to BATCH_LIST2 and BATCH_OUT2.
) CX START (BATCH_<LIST2 OUT2> LPT1) @LPB1 ↵	Start list, output, and LPT1 print queues at device LPB1.
) CX CONTINUE @LPB1 ↵	Continue the printer.

The following commands restore batch list and output print queues to the original queues, BATCH_LIST and BATCH_OUTPUT.

) CX BATCH_LIST/DEFAULT BATCH_IN2 ↵

) CX BATCH_OUTPUT/DEFAULT BATCH_OUT2 ↵

BATCH_OUTPUT

Sends all output files from a batch input queue to the specified print queue.

Format

CX BATCH_OUTPUT { *[batch–queue]* print–queue / /DEFAULT *[batch–queue]* }

where

batch–queue is the name of the batch input queue whose output files you are redirecting. If omitted, the batch queue is BATCH_INPUT.

print–queue is the name of the print queue you're associating with the specified batch queue or with BATCH_INPUT.

/DEFAULT tells the XBAT process to use the BATCH_OUTPUT print queue.

Description

The default batch input queue is BATCH_INPUT, and the default output print queue for all batch input queues is BATCH_OUTPUT.

This command changes the output print queue for the specified batch input queue. The print queue you specify must exist as a queue of type PRINT.

With the /DEFAULT switch, this command restores the output print queue for a batch input queue to its original default, BATCH_OUTPUT.

To use any batch input queue other than the default (BATCH_INPUT), a user must use the QBATCH command with the /QUEUE=queuename switch. Also, a user can specify a batch output file instead of the default output print queue with the QBATCH command /QOUTPUT=pathname switch. The output file specified may not be a queue.

Why Use It?

The BATCH_OUTPUT command gives you greater control over placement of batch output. You can use this command to send all batch output to a queue on a specific printer.

Example

The following commands create a new batch input queue and new batch list and output print queues, direct batch listings and output to the new print queues, and start the output and list print queues, as well as print queue LPT1, on the second line printer (LPB1).

) CX CREATE BATCH BATCH_IN2 ↵	Create batch input queue.
) CX OPEN BATCH_IN2 ↵	Open the queue for input.
) CX CREATE PRINT BATCH_LIST2 ↵	Create batch list print queue.
) CX CREATE PRINT BATCH_OUT2 ↵	Create batch output print queue.
) CX BATCH_LIST BATCH_IN2 BATCH_LIST2 ↵	
) CX BATCH_OUTPUT BATCH_IN2 BATCH_OUT2 ↵	Change list and output print queue names for new input queue to BATCH_LIST2 and BATCH_OUT2.
) CX START (BATCH_<LIST2 OUT2> LPT1) @LPB1 ↵	Start list, output, and LPT1 print queues at device LPB1.
) CX CONTINUE @LPB1 ↵	Continue the printer.

The following commands restore batch list and output print queues to the original queues, BATCH_LIST and BATCH_OUTPUT.

) CX BATCH_LIST/DEFAULT BATCH_IN2 ↵

) CX BATCH_OUTPUT/DEFAULT BATCH_OUT2 ↵

BINARY

Tells the XLPT process to enable or disable binary mode.

Format

CX BINARY @printername { filename / OFF }

where

@printername	is the pathname of a printer in :PER; for example, @CON27.
filename	is the cleanup filename. EXEC expects to find this file in directory :UTIL:FORMS.
OFF	disables binary mode.

Description

When binary mode is enabled on a printer, and users use the /BINARY switch to queue jobs to the printer, XLPT passes all characters between 0 and 376 octal directly to the printer without interpreting them. Binary printing is useful when users want to print on a device that interprets characters its own way, for example, a graphics printer. For such printers, you don't want XLPT to edit special characters that have meaning to the device; you want it to pass each character along as is.

Since it doesn't interpret control characters, XLPT cannot keep track of page numbers, line numbers, or any other data that requires character interpretation. So after a file has been printed in binary, the printer is left in an unknown state — perhaps with paper positioned in the middle of a page, for example.

Thus, you must supply the name of a cleanup file (filename) when you enable binary mode. The cleanup file must position the paper at the physical top of page (line 1) for the next user. To do this, the cleanup file need contain only a Form Feed (ASCII 14, CTRL–L). The XLPT process sends the cleanup file to the device when you enable binary mode and after the device prints a file in binary. The cleanup file must be in directory :UTIL:FORMS. For printers under the CEO system, you can create cleanup files easily when you define printers in the CEO system.

To have a file printed in binary mode, a CLI user must append the /BINARY switch to the QPRINT command, for example,

```
) QPRINT/QUEUE=LQP/BINARY MYFILE ↵
```

If binary mode is not enabled for the device associated with the specified queue, the printed output will report the error message

Binary mode not enabled

The printer that will process files in binary mode must be paused and idle before you can enable binary mode. After enabling binary mode, you must continue the printer.

There is one character that XLPT does interpret in binary mode: a character with a value of 377 octal. XLPT interprets or reads, but does not write 377 octal and any character (except 377 octal) that follows the first 377 octal. So if someone wants to pass a 377 octal to the printer in binary mode, the file must contain two sequential octal 377s. You can get around this by using the /PASSTHRU switch.

The /PASSTHRU switch causes XLPT to process all characters without special interpretation. This does away with the need to specify double 377 octal characters.

To see whether binary mode is enabled or disabled, use EXEC's SPOOLSTATUS command.

Why Use It?

Binary mode is required for letter–quality printers used by CEO users. It is also required when you need to have all characters printed precisely as they are in the file, without interpretation by the XLPT process.

If a device will be used exclusively in binary mode, you can put the BINARY command in the UP.CLI macro.

Example

```
) CX PAUSE @CON26 ↵
```

From EXEC: @CON26 PAUSED

```
) CX BINARY @CON26 CLEANUP_26 ↵
```

From EXEC: @CON26 Binary mode enabled

```
) CX CONT @CON26 ↵
.
.
.
) CX PAUSE @CON26 ↵
) CX BINARY @CON26 OFF ↵
) CX CONT @CON26 ↵
```

BRIEF

Tells EXEC to make its messages brief (opposite of VERBOSE).

Format

CX BRIEF *[devicename] [n]*

where

devicename is the name of the device. If the name is omitted, BATCH_INPUT is used as the device name.

n indicates the number of the device stream whose messages you want to make brief. If it is omitted, all batch streams associated with the device specified produce brief messages.

Description

When a stream or device accepts or processes a request, EXEC sends a message to the system console (@CON0 or @PMAP0). This message may be either *brief* or *verbose*. The EXEC BRIEF and VERBOSE commands determine the type of EXEC messages sent. Each time you issue a BRIEF or VERBOSE command, it overrides the current message setting. BRIEF is the default mode.

BRIEF messages include

- the device name and stream number;
- the job sequence number; and
- the user's username.

You can also suppress EXEC's time of day prompt with the command CX PROMPTS OFF. And you can suppress all status messages with the CX SILENCE command.

In addition to these two EXEC commands, you should consider using the CLI CHARACTERISTICS command with the /NRM switch if anyone uses the system console to do work. The /NRM (no receive messages) switch prevents noncritical message interruptions from a variety of programs (like XODIAC and CEO) that send messages to @CON0 or @PMAP0.

Why Use It?

You may find EXEC messages easier to read if they contain less information.

Example

```
) CX BRIEF BATCHQ ↵

.
.
.

From Pid 3 : (EXEC) BATCH_INPUT, STRM=1 SEQ=446 QPRI=127 USER=ROBIN

) CX BRIEF ↵

.
.
.

From Pid 3 : (EXEC) BATCH_INPUT, STRM=1 SEQ=447 QPRI=127 USER=F77

) CX BRIEF @LPB ↵

.
.
.

From Pid 3 : (EXEC) @LPB, STRM=1 SEQ=448 QPRI=127 USER=SALLY
```

CANCEL

Cancels waiting queue entries by sequence number or username.

Format

CX CANCEL { sequence–number | /USERNAME= }

where

sequence–number	is the sequence number of the queue entry to be cancelled.
/USERNAME=	is a switch specifying cancellation of all inactive jobs for a given user.

Description

The CANCEL command cancels the specified queue entry or a specified user's inactive jobs. It doesn't work for active jobs; use the FLUSH command for these.

The CLI command QDISPLAY lists queue entries and their queue sequence numbers. Active entries are marked with an asterisk (*) in the display. Entries that the operator has cancelled appear with an F flag in the display.

After you cancel an entry, the batch output file or the printed output file will show the message *Cancelled by Operator.* For batch requests, the temporary batch input file remains in the user's directory.

Instead of cancelling an entry yourself, you might send a message to the user, asking him or her to cancel it with the CLI command QCANCEL.

Why Use It?

There may be times when you don't want requests to remain queued. For example, a print queue may contain a job that no longer needs to be printed; or, you may want to cancel a large print job and re–queue it as a batch job to be printed at night.

Example

) QD ↵

BATCH_INPUT BATCH Open

535 JACK :UDD:JACK:F77:?040.CLI.004.JOB
536 JILL :UDD:JILL:JILLSFILE
537 JACK :UDD:JACK:JACKSFILE

.
.

) CX CAN 535 ↵

) QD ↵

BATCH_INPUT BATCH Open

536 JILL :UDD:JILL:JILLSFILE
537 JACK :UDD:JACK:JACKSFILE.
.
.

Here, the CANCEL command cancelled the request with sequence number 535. The printer became active because it was printing the batch output file with the message *Cancelled by Operator.*

In the next example, both of Jack's print jobs are cancelled with the /USER= switch.

) QD ↵

BATCH_INPUT BATCH Open

535 JACK :UDD:JACK:F77:?040.CLI.004.JOB
536 JILL :UDD:JILL:JILLSFILE
537 JACK :UDD:JACK:JACKSFILE

.
.
.

) CX CAN/USER=JACK ↵

) QD ↵

BATCH_INPUT BATCH Open

536 JILL :UDD:JILL:JILLSFILE.

CLOSE

Closes the specified queue to user requests (opposite of OPEN).

Format

CX CLOSE queuename

where

queuename is the queue you want to close.

Description

The CLOSE command closes the specified queue, which prevents users from submitting more requests to the queue. Once you close the queue, users will get *Closed* error messages when they submit requests to the queue. The queue continues to process requests within it, but accepts no new requests.

To reopen a queue, you must use the EXEC OPEN command.

The CLI command QDISPLAY will tell you which queues are open and which are closed.

Why Use It?

Occasionally, you may want a queue to stop accepting requests; perhaps if you want to reorganize it or change it, or if the device involved needs servicing. The CLOSE command prevents the queue from accepting any more requests. If you want, you can then use CX FLUSH followed by CX PURGE to empty the entire queue of all active and inactive requests.

It is not always necessary to close a queue in order to service a device such as a printer. You might want to use the CX PAUSE command to pause the queue. This stops requests from being printed until the device is repaired, but still allows users to submit requests. Requests submitted while the queue is paused will be printed when the device is continued later.

Example

```
) CX CLOSE LPT ↵

) QPRINT MYFILE ↵

Warning: Queue is not open
)
```

CONSOLESTATUS

Displays status of each terminal enabled by EXEC.

Format

CX CONSOLESTATUS *[@consolename]*

where

@consolename is the device name for the terminal, preceded by @ to specify the peripherals directory; for example, @CON44.

Description

If you omit an argument, the CONSOLESTATUS command displays the status of all enabled terminals. To check a specific terminal, give its name as an argument.

For each active terminal, EXEC displays the username, terminal name, and process ID. It also tells you whether the terminal will be disabled when the current user logs off and whether a user is in the process of logging on or off.

Why Use It?

The CONSOLESTATUS command can be used to find out if a particular terminal is enabled, or if someone is logged on to an enabled terminal. The CONSOLESTATUS command identifies terminals on which someone is logging on or off, and it can help you identify active terminal lines.

Example

```
) CX CONSOLES ↵

... (status information on all enabled terminals) ...

) CX CONSOLES @CON45 ↵

@CON45 Enabled, Logon Tries = 5, Continue, Pid: 36, User = SAM

) CX CONSOLES @CON4(6 7 8) ↵

@CON46 Enabled, Logon Tries = 5, Continue, Not logged on
@CON47 Enabled, Logon Tries = 5, Continue, Log on/off in progress
Warning: Console unknown to EXEC
)
```

These messages indicate an enabled terminal in use, not in use, in logon/logoff transition, and a terminal not enabled by EXEC. The parentheses enclosing 6 7 8 in the last command are a CLI feature to help you write compact command lines.

CONTINUE

Directs one or more device streams to begin or resume processing (opposite of PAUSE).

Format

CX CONTINUE *[devicename][n]*

where

devicename is the name of the device that you're continuing. If the device name is omitted, the command defaults to BATCH_INPUT.

n indicates the number of the stream that you're continuing. If it is omitted, EXEC continues all batch streams associated with the device specified.

Description

The CONTINUE command continues a paused device (all streams, or specified stream).

If the command works, subsequent STATUS commands will not show the *Paused* message.

Why Use It?

The CONTINUE command is needed to restore normal processing whenever you have paused a specified stream or device. It is also needed after you have started a device on one or more queues. The macro UP.CLI issues several CONTINUE commands as it brings up the multiuser environment.

Example

```
) CX PAUSE @CON26 ↵

From EXEC: @CON26 PAUSED

) CX BINARY @CON26 CLEANUP_26 ↵

From EXEC: @CON26 Binary mode enabled

) CX CONT @CON26 ↵
```

CPL

Changes the number of characters per line for a device.

Format

CX CPL devicename number

where

devicename — is the name of a device.

number — is the number of characters per line. It must be an integer between 16 and 255.

Description

The CPL command sets a new number of characters per line (CPL) for this device. It overrides any previous VSGEN, CHARACTERISTICS/CPL, or EXEC CPL settings. The new CPL remains until you change it with EXEC's CPL or bring down EXEC's XLPT cooperative process.

To change a CPL setting:

- Type CX PAUSE to the device.
- When EXEC says that the device is paused, type the desired CX CPL command.
- Type CX CONTINUE to the device.

The CPL command does nothing on a device if an EXEC DEFAULTFORMS or FORMS command is in effect on the device.

Why Use It?

The default number of characters per line is 80.

On any printer, if a line to be printed exceeds the maximum number of characters per line it will be truncated to CPL on output unless the user applied the /FOLDLONGLINES switch to the QPRINT command. Having lines truncated is undesirable, and having them folded is often messy. To avoid both, you will generally want the longest possible line to print as in the file — and most printing paper has space for at least 85 characters per line (80 column) or 136 characters per line (132 column). To have more than 80 characters printed, you must use the CPL command.

After deciding on a good CPL for each printer, you can specify it in the UP.CLI macro before continuing the printer.

Example

) CX PAUSE @LPB1 ↵

) CX CPL @LPB1 136 ↵

) CX CONT @LPB1 ↵

To put the CPL command in the UP.CLI macro, you'd simply insert the line

CX CPL @LPB1 136

before the EXEC CONTINUE command for @LPB1.

CREATE

Creates a queue (opposite of DELETE).

Format

CX CREATE *[/STREAMS=n]* queue–type queuename

where

/STREAMS=n indicates the number of streams you're creating for this queue (the default is 1 and the range is 0 through 10). This switch is valid only for batch queues.

queue–type specifies the type of queue you're creating.

queuename is the name of the new queue.

Description

Use the CREATE command to create queues. The new queue name can contain any legal filename characters, but to access it with a CLI Q–series command (such as QPRINT or QPLOT) without using a /QUEUE= switch, the queue names must be

Queue Type	Queue Name	Accessed by CLI Command
BATCH	BATCH_INPUT	QBATCH
FTA	FTQ	QFTA
PLOT	PLT	QPLOT
PRINT	LPT	QPRINT
HAMLET	HAMQ	QSUBMIT
SNA	SNA	QSNA
MOUNT	MOUNTQ	MOUNT

To access a queue with any other name, users must append the /QUEUE= switch. For example, for a queue named LPT1:

```
) QPRINT/QUEUE=LPT1 MYFILE ↵
```

Users can submit jobs to a queue in three ways. They can

- Specify the queue pathname as a listing file with the /L= switch; for example, WRITE/L=@LPT1 Hello and press NEW LINE;
- Specify the queue pathname as an output destination file in a CLI command; for example, COPY @LPT1 MYFILE and press NEW LINE; or
- Explicitly type a CLI QPRINT, QPLOT, QPUNCH, QFTA, or QSUBMIT (for HASP II/HAMLET) command; for example, QPRINT MYFILE and press NEW LINE.

CREATE (Continued)

One CREATE command is needed for each queue in your system, except for BATCH_INPUT, BATCH_OUTPUT, BATCH_LIST, and MOUNTQ, which are automatically created by QCMP. Then for the new queue, the OPEN, START and CONTINUE commands are needed to make the queue usable. Users cannot submit jobs to the queue until it is open, and jobs won't be processed until the queue has been started on a device and the device has been continued.

The CREATE and OPEN commands are generally issued only once, to set up the queue in :QUEUE and :PER. The START and CONTINUE commands, which are usually found in the UP.CLI macro, can then be used to allow the jobs in the queue to be processed.

See the information on creating the multiuser environment in the *Installing* manual for your particular operating system. This information has the reader create and open the needed queues, then edit START and CONTINUE commands in the UP.CLI macro.

Why Use It?

EXEC, as shipped, contains one batch queue, a mount queue, a batch output print queue, and a batch list print queue. Most systems also need a print queue (default LPT) for QPRINT requests; queues for such things as laser printing and restricted batch processing may also be desired. These queues must be created with the CREATE command.

Example

The following commands create, open, and start the default print queue named LPT for the printer LPB:

```
) CX CREATE PRINT LPT ↵
) CX OPEN LPT ↵
) CX START LPT @LPB ↵
) CX CONTINUE @LPB ↵
```

The following example shows how to create, open, and start for a print queue named PRINTER:

```
) QPRINT/QUEUE=PRINTER MYFILE ↵
Warning: Queue does not exist, QPRINT/QUEUE=PRINTER, MYFILE

) CX CREATE PRINT PRINTER ↵
) CX OPEN PRINTER ↵
) CX START PRINTER @LPB ↵
) CX CONTINUE @LPB ↵
) QPRINT/QUEUE=PRINTER MYFILE ↵
Queued, Sequence number = 1. Qpriority = 127

)
```

The following command lines create, open and start a batch input queue with eight streams.

```
) CX CREATE/STREAMS=8 BATCH BATCH2 ↵
) CX OPEN BATCH2 ↵
) CX CONTINUE BATCH2 ↵
```

DEFAULTFORMS

Sets the characteristics of the default form for a print device.

Format

CX DEFAULTFORMS @printername *[form–name]*

where

@printername is the pathname of a printer in :PER; for example, @LPB1.

form–name is the name of a file containing formatting commands for printed output. EXEC expects to find this file in directory :UTIL:FORMS.

Description

The DEFAULTFORMS command sets the default form for a device. All files printed on this device will be printed per this form unless a user asks for a special form with the /FORMS= switch.

The new form specifications must be placed in a form file in :UTIL:FORMS. You, or a user, can put printing directives in the form file with the FCU utility, which is explained in the manual *Using the CLI, (AOS/VS and AOS/VS II)*.

If you omit a form–name argument, EXEC uses the following default characteristics for printing files:

- The current setting for lines per page (LPP command); default is 66;
- The current setting for characters per line (CPL command); default is 80;
- The top of form is line 1 if LPP is less than 7; otherwise, the top of form is line 4;
- The bottom of form is the number of lines per page (LPP).

If you use the DEFAULTFORMS command to set lines per page or characters per line, you cannot use EXEC's LPP or CPL command to change these while the new DEFAULTFORM is in effect. Instead you must reissue DEFAULTFORMS without a form–name argument; this restores the standard form, allowing LPP or CPL commands to work.

EXEC's SPOOLSTATUS command will tell you if nonstandard DEFAULTFORMS values are in effect.

As with any device parameter change, the device must be paused before EXEC will accept the command.

Why Use It?

For specific nonstandard printing jobs, you'd use EXEC's FORMS command. If most or all of a printer's work will be on nonstandard forms, you might want to make the nonstandard form the default form for this printer.

Example

```
) CX PAUSE @LPB1 ↵

) CX DEFAULTFORMS @LPB1 STANDARD_LPB1 ↵

) CX CONTINUE @LPB1 ↵
```

This sequence sets the standard form for @LPB1 to the specifications in the file :UTIL:FORMS:STANDARD_LPB1 (built with the FCU program). Later to restore the standard form to LPB1, you'd type

```
) CX PAUSE @LPB1 ↵

) CX DEFAULT @LPB1 ↵

) CX CONT @LPB1 ↵
```

DELETE

Deletes a queue (opposite of CREATE).

Format

CX DELETE queuename

where

queuename is the name of the queue you want to delete.

Description

The DELETE command deletes the specified queue. The queue must be closed, stopped, and empty before EXEC will accept this command. Use the CX CLOSE, STOP, and PURGE commands to do this.

Why Use It?

Use DELETE to get rid of queues no longer needed on your system.

Example

The following commands close a queue named printer, disassociate the queue from whatever devices are processing it, purge the queue of all its entries, and finally delete the queue entirely.

) CX CLOSE PRINTER ↵

) CX STOP PRINTER ↵

) CX PURGE PRINTER ↵

) CX DELETE PRINTER ↵

DISABLE

Prevents logon from a terminal or all terminals (opposite of ENABLE).

Format

CX DISABLE { @consolename / /ALL }

where

@consolename	is the device name for the terminal; it must begin with @; for example, @CON40.
/ALL	tells EXEC to disable all enabled terminals.

Description

The DISABLE command removes the EXEC logon capability provided by the ENABLE command. It prevents users from logging on at the specified terminal. If a user is logged on a terminal, EXEC will not log him or her off. Instead EXEC will wait until the user has logged off before implementing the DISABLE command and will display a *Will be disabled* message on the system console.

You can undo a DISABLE command to an active terminal by issuing an ENABLE command to that terminal before the user logs off.

After EXEC disables a terminal, it displays the message

— *sysid Console DISABLED from logging on* —

on that terminal.

Why Use It?

Often you will want to shut the system down or release a terminal from EXEC so that another program (like DG/SNA) can use it. In either case, you don't want people to log on via EXEC. Use the DISABLE command to handle either situation.

Another reason to use this command is to disable unterminated lines. The operating system does not detect unterminated lines. However, you may explicitly disable them from EXEC and then assign them so that no program can open them.

Example

) CX DISABLE @CON2 ↵

) CX DISABLE @CON4(7 8 9) ↵

Console will be disabled
Warning: Console unknown to EXEC
Warning: File does not exist

) CX ENABLE @CON47 ↵

Console enabled, @CON47

These commands attempt to disable CON2, CON47, CON48, and CON49 and then undo the disable on CON47. EXEC returned no message from the first command, meaning that CON2 was already disabled. The other messages mean that CON47 was in use, that CON48 wasn't enabled, and that CON49 wasn't specified to VSGEN.

) CX DISABLE/ALL ↵

All consoles will be disabled.

This command will cause EXEC to disable all terminals as each user logs off.

DISMOUNTED

Tells EXEC that you have physically dismounted the tape(s) (opposite of MOUNTED).

Format

CX DISMOUNTED [*mount–ID* | *@tapeunit*]

where

mount–ID is the mount identifier: an integer shown by EXEC for each MOUNT request and displayed by EXEC's MOUNTSTATUS command.

@tapeunit is the pathname of a magnetic tape unit; for example, @MTB1 or @MTC0.

Description

The DISMOUNTED command tells EXEC that you have physically removed the tape from the unit. EXEC deletes the user's link name to the tape unit and restores the unit's ACL to the setting it had before the mount.

If you omit arguments, the command applies to all tape volumes in the current dismount request. If you give a mount identifier (mount–ID), this tells EXEC that all tapes associated with the specified request have been dismounted; handy when you want to specify a request other than the current request. The /MID switch is most useful when you have multiple requests. EXEC deals with dismount requests on a first–come, first–served basis. Thus, EXEC is likely to dismount the first request it sees, which may not be the request you want. Using the /MID switch ensures that EXEC acts on the proper request.

If you give a tape unit name, this tells EXEC that the tape on a unit has been dismounted. This form is meant for situations where you use the PREMOUNT command on a volume, and then change your mind and want to dismount it. It's also useful when you have mounted two units for a user and want to dismount one of them so another user can use it.

If SYSLOG logging is on (see Chapter 11), the log file will record the elapsed time that the user had the tape mounted.

EXEC prompts you to type the DISMOUNTED command

- When a user who has a tape mounted types the CLI command DISMOUNT;
- When a user who has a tape mounted logs off without typing the CLI command DISMOUNT;
- When a user's I/O is complete after you've complied with his or her implicit mount request (for example, typing LOAD_II/V @LMT:GP0076:MY_FILESET PROG+ and pressing NEW LINE).

DISMOUNTED (Continued)

When EXEC prompts you to type DISMOUNTED, it may also display a *Request is* line with additional information from the user. The *Request* line will not appear in an implicit mount request. It will also not appear if the user program terminated before the tape was dismounted or if the user omitted a comment when he or she typed the CLI DISMOUNT command.

Be sure to dismount the tape physically before issuing the DISMOUNTED command to prevent another user from accidentally writing to it.

Why Use It?

You must issue DISMOUNTED when EXEC prompts for it to release the tape unit(s) from EXEC's mount mechanism.

Use CX DISMOUNTED with the devicename when you have premounted a volume, but have dismounted it before it has been used.

Example

An unlabeled tape user asks for dismount:

From Pid 257 : (XMNT) 23-Jul-1993 10:38:17
****
Unit Dismount Request
****
MID = 67 *USER= SCOTT*
User Pid = 47 *Requestor Pid = 63*
Units: *MTB0*
Dismount Unit: *MTB0*
User Message: *Please mark the tape as mine.*

Respond: *CX DISMOUNTED @[unitname]*

Remove the tape from the unit and type

) CX DISMOUNTED ↵

A labeled tape user asks for a dismount:

From Pid 257 : (XMNT) 23–Jul–1993 10:38:17

```
**********
Unit Dismount Request
**********
MID = 67              USER= SCOTT
User Pid = 47      Requestor Pid = 63
Volumes:              VOL0, VOL1
Units:                MTB0/VOL0, MTB1/VOL1
Dismount Unit:        MTB0, MTB1
User Message:         Please mark the tapes as mine.

Respond:              CX DISMOUNTED @[unitname]
```

Remove the user's tapes from MTB0 and MTB1. Then type

```
) CX DISMOUNTED ↵
```

If you premount a tape, then type

```
) CX PREMOUNT @MTB0 DB0067 DATABASE ↵
```

and then change your mind about the premount, remove the tape from the unit and type

```
) CX DISMOUNTED @MTB0 ↵
```

ELONGATE

Turns elongated printing on or off on an LP2 or TP2 printer.

Format

CX ELONGATE @printername { ON / OFF }

where

@printername is the pathname of a print device in :PER; it must be the pathname of a DASHER® LP2 line printer or DASHER TP2 terminal printer.

Description

DASHER LP2 and TP2 printers feature elongated printing that allows you to vary character width. The ELONGATE command turns elongated printing for either of these on or off. If you attempt to issue the ELONGATE command to any other type of printer, EXEC responds with the message

Wrong device type for command.

The device must be paused before you issue the ELONGATE command. After you change the setting, type the CX CONTINUE command and press NEW LINE.

Why Use It?

If one of these DASHER devices is under control of EXEC's XLPT process, you must use the ELONGATE command to get elongated printing on it.

Example

```
) CX PAUSE @LPC ↵

) CX ELONGATE @LPC ON ↵

) CX CONTINUE @LPC ↵

From Pid 3 : (EXEC) @LPC [Idle]
```

ENABLE

Allows logon and/or sets logon parameters (opposite of DISABLE).

Format

CX { ENABLE *[/TRIES=n]* [*/STOP* / */CONTINUE*] *[/FORCE] [/BRIEF]* @consolename
CX { ENABLE/ALL *[/TRIES=n]* [*/STOP* / */CONTINUE*] *[/FORCE] [/BRIEF]*

where

/TRIES=n tells EXEC how many logon tries anyone can make before it locks the terminal for 10 seconds (if /CONTINUE is in effect) or disables the terminal (if /STOP is in effect). The valid range is 1 through 10. The default value is 5.

/STOP tells EXEC to disable the terminal if anyone fails to log on in n tries (default is 5). On a modem line, it also breaks the connection. No one can log on the terminal until it is enabled again.

/CONTINUE tells EXEC to leave the terminal enabled after someone has failed in *n* tries to log on. After *n* failed tries, EXEC displays *Too many attempts* and locks the terminal for 10 seconds. On a modem line, it also breaks the connection. Continue is the default value if you do not specify /STOP.

/FORCE tells EXEC to force the values set by the other switches onto the terminal whether or not the terminal is enabled. The logon values will take effect the next time someone tries to log on (as usual). The imposition of new values is not visible to anyone who is logged on. If you omit /FORCE and a terminal is enabled, EXEC returns a *Console already enabled* error.

/BRIEF CLI32 only. Tells EXEC to suppress enable messages for each console, reporting only errors.

@consolename is the device name for the terminal, preceded by @; for example, @CON78. Or, you can create a file containing console names and issue CX ENABLE ([filename]) to enable the specified terminals. Each line but the last in filename must consist of the consolename followed by a space and an ampersand. Omit the space preceding the ampersand in the last line. For example:

```
@con33 &
@con34 &
...
@con35&
```

/ALL tells EXEC to enable all terminals that were identified to VSGEN, and virtual terminals. You must choose either /ALL or the specific terminal (@consolename).

ENABLE (Continued)

Description

The ENABLE command gives the specified terminal — or all terminals — logon capability with specific logon parameters. A user can then log on the enabled terminal(s) via EXEC and use the operating system.

If you omit logon parameters (/STOP, /TRIES, etc.), EXEC uses the default parameters. The defaults are /TRIES=5 and CONTINUE.

When an ENABLE @consolename command succeeds (unless you use the /BRIEF switch), EXEC displays on the system console

Console Enabled, @CONn

For an ENABLE/ALL command, EXEC displays

Enabling all consoles

... (EXEC displays a list of terminals as they are enabled) ...

Enable all complete, n consoles enabled

In either case, if any terminal is in use by another terminal–managing program, EXEC displays an error message. EXEC also returns an error if the terminal is already enabled — unless you include the /FORCE switch in the ENABLE command. With the ENABLE/ALL command, EXEC continues enabling after an error.

After being enabled, each terminal (if on line) shows the message

**** sysid Press NEW LINE to begin logging on ****

Or it shows what is in the tailored logon screen.

You cannot enable CON1; this terminal does not exist in AOS/VS or AOS/VS II.

To discover the ENABLE values for a terminal, use EXEC's CONSOLESTATUS command.

Why Use It?

The ENABLE command is needed to give most users access to the system. Usually, the UP.CLI macro enables the desired terminals via ENABLE/ALL or multiple ENABLE commands.

The switches /STOP and /TRIES= are useful if you're concerned with security over modem lines or with local terminals. Breaking into a system whose remote terminals are enabled with /TRIES=1/STOP is very difficult. You might set up the secure ENABLE values in a special macro — to run at night — with the /FORCE switch to make sure values are enforced. Before running the macro, you could notify users with the BROADCAST macro.

After running the special UP.CLI macro, you might run it again (perhaps every hour) to re–enable terminals that had been disabled by failed logon tries. Running such an UP macro would give legitimate users a chance to log on.

Example

) CX ENABLE @CON(3,4) ↵

Console enabled, @CON3

) CX ENABLE/TRIES=3/STOP @CON3 ↵

Console already enabled, @CON3

) CX ENABLE/TRIES=3/FORCE/STOP @CON3 ↵

New logon values in effect, @CON3

) CX ENABLE @CON(4,5) ↵

Console already enabled, @CON4
From EXEC: Console enabled, @CON5

) CX ENABLE/ALL ↵

Enabling all consoles
Enable all complete, 40 consoles enabled

EVEN

Turns even pagination on or off for a printer.

Format

CX EVEN @printername { ON / OFF }

where

@printername is the pathname of a printer in :PER; for example, @LPB.

Description

The EVEN command turns even pagination on or off for the specified device. The default setting is ON. You can check the current setting with EXEC's SPOOLSTATUS command. The device must be paused before EXEC will accept the command.

When EVEN is ON, XLPT adds a blank page to the end of files with an odd number of pages. Adding this extra blank sheet allows all header pages to print on the same fold of fanfold paper; the header page will always be on top.

When EVEN is OFF, XLPT prints queued files just as they are, one after the other, with no intervening blank pages.

Why Use It?

Generally, turn EVEN off for sheet–feed printers (laser, etc.) and on for fanfold printers (line printers).

Example

```
) CX PAUSE @LPB1 ↵

) CX EVEN @LPB1 OFF ↵

) CX CONTINUE @LPB1 ↵

From Pid 3 : (EXEC) @LPB1 [Idle]
```

FLUSH

Flushes (terminates) the active request(s) in a device stream or streams.

Format

CX FLUSH *[devicename] [n]*

where

devicename is the name of the device whose streams you're flushing. BATCH_INPUT is the default device name if the device name argument is omitted.

n indicates the number of the stream you're flushing. EXEC flushes all streams of the device specified if you omit this argument. This argument is required if you don't specify a device name.

Description

The FLUSH command tells EXEC to stop processing the active request in a device stream, or the active request a specified device is processing. EXEC then starts the next request (if any).

For each flushed job, the batch output or printed file on the printer will show a *Terminated by Operator* message. (On a slow printer such as a letter–quality printer, it may take a moment for the buffer to empty and the printer to stop.)

Why Use It?

Sometimes you will want to stop an active job — for example, to

- terminate a job that is looping;
- stop printing a file that you don't want printed; or
- empty queues so you can shut down EXEC.

The FLUSH command is the easiest way to kill an active request. If you don't want the next queued job to be processed, issue a PAUSE command before FLUSH.

Example

From Pid 3 : (EXEC) BATCH_INPUT, STRM=1, SEQ=582 QPR=127 USER=ROBIN

) CX FLUSH 1 ↵

From Pid 3 : (EXEC) BATCH_INPUT, STRM=1 Flushing Current Job at User Request
From Pid 3 : (EXEC) BATCH_INPUT, STRM=1 SEQ=583 QPR=127 USER=MARTIN

) CX FLUSH BATCH_2 (2,3) ↵

From Pid 3 : (EXEC) BATCH2_2 Flushing Current Job at User Request
Warning: No current job

) CX FLUSH @LPB ↵

From Pid 3 : (EXEC) @LPB Flushing Current Job at User Request

This command sequence flushes stream 1 of the BATCH_INPUT queue which then accepts the next request. Then in one FLUSH command, it attempts to flush streams 2 and 3 of the BATCH2 queue. Finally it flushes the current @LPB request.

You can also use the FLUSH command with MOUNT and FTA jobs. Because FTA is a multistream cooperative, you must specify a device name and a stream number when you flush FTA jobs, as shown below.

) CX FLUSH @FTA_EXEC 4 ↵

FORMS

Tells a printer to use special forms for printing.

Format

CX FORMS @printername *[form–name]*

where

@printername is the pathname of a printing device entry in :PER; for example, @LPB1.

form–name is the name of a file containing formatting commands for printed output. EXEC expects to find this file in directory :UTIL:FORMS.

Description

The FORMS command directs a printer to process all requests that specify the special form *form–name*, the name of a special form, like a paycheck form for a line printer. Someone must have created the form file in the :UTIL:FORMS directory and tailored it with the FCU program, which is described *Using the CLI (AOS/VS and AOS/VS II).*

A user can request a special form by appending the /FORMS=form–name switch to a CLI QPRINT or QPLOT command. After a user does this, EXEC retains the request in the output queue until you type a FORMS command that specifies the form–name. To see which forms await special forms handling, type

) QDISPLAY/V ↵

When you decide to process output requests that need a special form on any device, follow these steps:

1. Pause the printer by using the CX PAUSE @printername format. Wait until EXEC tells you that the printer is paused.
2. Insert the desired forms in the printer.
3. Type the command CX FORMS @printername form–name, to indicate that the new form has been loaded.
4. Continue the printer (using the format CX CONTINUE @printername).

The printer then starts printing the forms. It prints only those files submitted with QPRINT/FORMS=form–name; other print requests to that printer wait in the queue.

When the printer finishes the special form requests, it becomes idle.

5. Now, type CX PAUSE @printername.
6. Reinsert the original forms and type CX FORMS @printername.
7. Continue the printer with CX CONTINUE @printername.

The printer now processes output requests submitted without a /FORMS= switch.

FORMS (Continued)

Why Use It?

Many installations use special printing forms; for example, for paychecks or invoices. The FORMS command allows you to dedicate a printer to printing all user requests that specify a special form. (You could use the DEFAULTFORMS command instead if you wanted all requests to the printer printed the same way.)

Example

From Pid 15 : Ready to print paychecks.

) SEND 15 Ok type QPRINT/FORMS=PAYCHECK_FORM CHECKS ↵

From Pid 15 : Ok typed QPRINT command.

) CX PAUSE @LPB1 ↵

From EXEC time

... (someone puts paycheck forms in printer) ...

) CX FORMS @LPB1 PAYCHECKS ↵

) CX CONTINUE @LPB1 ↵

... (paycheck printing occurs) ...

From Pid 3 : (EXEC) @LPB1 [Idle]

) CX PAUSE @LPB1 ↵

From Pid 3 : (EXEC) @LPB1 [Idle]

... (someone puts standard paper in printer) ...

) CX FORMS @LPB1 ↵

) CX CONTINUE @LPB1 ↵

From Pid 3 : (EXEC) @LPB1 [Idle]

Here the operator receives a message that payroll is ready to print paychecks. The operator sends payroll the QPRINT command syntax. Payroll types and verifies the QPRINT command, and the operator pauses the printer. Then someone puts the blank paycheck forms in the printer and continues the printer. The operator sets the PAYCHECK printing specification with EXEC's FORMS command, and the checks are printed.

After printing, the operator receives the *[Idle]* message and pauses the printer again. Someone puts standard paper in the printer. The operator sets standard @LPB form specification, and the printer resumes processing standard printing requests.

HALT

Terminates EXEC.

Format

CX HALT

Description

The HALT command has no switches or arguments. HALT directs EXEC to terminate immediately. In order to issue the HALT command, you must have the same username as EXEC (almost always OP) or you must have the System Manager privilege. You do not, however, have to be PID 2 to terminate EXEC with the HALT command.

Why Use It?

HALT terminates EXEC without creating a break file. For this reason, it is preferable to use HALT rather than the CLI TERMINATE command. You probably won't ever issue the HALT command directly. Instead, if you want to stop EXEC, use your system's DOWN.CLI macro or stop the operating system from the SMI (which then issues the EXEC HALT command).

As mentioned earlier, warn users with a broadcast message before you bring down the EXEC; otherwise EXEC logs users off in the middle of their work.

When you issue CX HALT, EXEC sends the following message to CON0:

From Pid 3: (EXEC) Terminating on HALT command.

Example

) BROADCAST Please log off. EXEC coming down in 2 minutes. ↵

Then execute the DOWN.CLI macro.

HEADERS

Changes the number of header sheets before each printed job.

Format

CX HEADERS @printername { 0 | 1 | 2 }

where

@printername is the pathname of a printing device in :PER; for example, @LPB1.

0, 1, or 2 is the number of header sheets you want.

Description

The HEADERS command sets the number of header sheets to be printed before the body of each printing request. By default, EXEC provides one header sheet per request. The header sheet gives the following information

- destination (username by default);
- username;
- queue name;
- device name;
- sequence number;
- qpriority;
- characters per line;
- lines per page;
- number of copies requested;
- MAPPER file;
- page limit;
- switches applied by the user (for example, /NOTIFY, /FOLDLONGLINES);
- date(s) the file was created, queued, and printed;
- pathname of the file;
- filename (first eleven characters);
- system identifier (SYSID); and
- revision of the operating system and XLPT program.

Figure 3–2 shows a sample header sheet.

```
+0000000000000000000000000000000000000000000000000000000000000000+

                $  $    $  $$$$    $    $
                $  $    $  $    $  $    $
                $  $    $  $    $   $ $
                $  $    $  $    $     $
                $  $    $  $    $     $
          $     $  $    $  $    $     $
           $$$     $$$   $$$$      $

        DEST=JUDY USER=JUDY QUEUE=LPT DEVICE=@LPB
      SEQ=5388 QPRI=127 CPL=80 LPP=65 COPIES=1 PAGES=145
                  /NOTIFY, /FOLDLONGLINES

                CREATED:  23-JUL-93 11:28:56
                ENQUEUED: 23-JUL-93 11:39:54
                PRINTING: 23-JUL-93 11:39:56

                      PATH=:MACROS:FF

                     $$$$$  $$$$$
                     $      $
                     $      $
                     $$$$   $$$$
                     $      $
                     $      $
                     $      $

+0000000000000000000000000000000000000000000000000000000000000000+



                           DOC4


                    AOS/VS II RELEASE 3.00.00.00
                  AOS/VS II XLPT-32 REVISION 3.00.00.00
```

Figure 3–2 Sample Header Page

HEADERS (Continued)

EXEC's SPOOLSTATUS command describes the number of headers set for a device.

As with CPL and other printer commands, you must pause the printer before EXEC will accept the command; you must continue the device afterward.

Why Use It?

On a slow printing device (like a letter–quality printer), omitting a header page can speed up processing.

For standard printers: some installations prefer 2 header pages to provide more separation between printing jobs. Others want to save paper, thus specify 0 headers. Generally, the default of 1 page works best.

Example

```
) CX PAUSE @LPB1 ↵

) CX HEADERS @LPB1 2 ↵

) CX CONTINUE @LPB1 ↵

From Pid 3 : (EXEC) @LPB1 [Idle]

)
```

HOLD

Prevents a job, or all jobs for a specified user, from running until you issue an UNHOLD or CANCEL command.

Format

CX HOLD { sequence–number / /USERNAME= }

where

sequence–number	is the sequence number of a job in the queue, displayed by the CLI command QDISPLAY.
/USERNAME=	is the name of the user who submitted the job.

Description

The HOLD command prevents a job — or all jobs of a specified user — from running until you issue an EXEC UNHOLD or CANCEL command. If you specify a sequence number, EXEC holds that sequence number. If you specify a username, EXEC holds all queue entries with that user's name.

The CLI QDISPLAY command lists queue entries and their sequence numbers. Active entries appear with an asterisk (*) in the display. (You cannot use HOLD with an active entry; to stop it, use the CX FLUSH command.) Entries held by the system operator appear with an E in the display. Entries held by users appear with H in the display.

Why Use It?

Sometimes you will have doubts about a request — perhaps you may not want to cancel it, but want to think about it. Use HOLD in such situations.

Example

```
) CX HOLD 26 ↵

) CX HOLD/USERNAME=OP ↵
```

These commands hold a request with sequence number 26 and those submitted by OP.

LIMIT

Prevents a stream, queue, or device from running jobs with page or CPU–time limits greater than those specified.

Format

CX LIMIT [*devicename*][*n*] [*hh:mm:ss* | *pages*]

where

devicename is the name of the device whose CPU use, or number of pages to be printed, you're limiting. If omitted, the default is BATCH_INPUT and you must include n.

n indicates the number of the stream you're limiting. If omitted, EXEC limits all streams for the device indicated.

hh:mm:ss is the maximum amount of CPU time in hours, minutes, and seconds that a job in the stream(s) can use. The legal range is 0:00:02 to 36:24:30. (Specify an even number for seconds; otherwise the system subtracts one second.) You must specify a stream as well as a time. If you omit the time or if you specify 0:00:00, the default CPU limit is 36:24:30.

pages is the maximum number of pages that a request may print. The number must be between 0 and 65,535. Users can specify a smaller limit for their jobs with the /PAGES switch.

Description

The LIMIT command sets limits on CPU time (for specified streams, print queues, or devices) or printed pages (for printing requests).

If limiting is not in effect, EXEC processes all jobs on a first–come, first–served basis, based on priority. Each stream can use up to 36:24:30 of CPU time; each printer can print up to 65,535 pages per request.

You can undo the effect of the LIMIT command with EXEC's UNLIMIT command. Any LIMIT command remains in effect until you change it, undo it with UNLIMIT, or until EXEC terminates.

To find out if limiting is enabled, and the maximum CPU limit, use EXEC's STATUS command.

Why Use It?

Limiting a printer can be useful if you have more than one printer. For example, assume you have two printers, LPB and LPB1. You can restrict one printer to small printing jobs:

```
) CX LIMIT @LPB 40 ↵
```

and leave the other printer unlimited. Users can send small printing jobs to @LPB (QPRINT command) and large ones to LPB1 (QPRINT/QUEUE=LPT1 command).

The LIMIT command gives your site control over requests before they are processed. It can help make the system more efficient by dedicating queues, streams, or devices to small or medium jobs. You can use it in conjunction with EXEC's QPRIORITY command to fine tune your system's queue processing, perhaps based on the time of day. You can change the emphasis quickly; for example, you can impose limiting only during the day, and lift it for big batch runs at night.

Also, the LIMIT command can help prevent program errors from slowing down the system; for example, it can prevent looping batch or runaway printing jobs from monopolizing the CPU or printer.

Examples

```
) BROADCAST streams 1–3 of BATCH_INPUT limited to 1:00 & ↵
&) of CPU time. Stream 4 unlimited. & ↵
&) Big batch users please use /CPU=0:10:0 to submit jobs. ↵

) CX LIMIT (1,2,3) 1:00 ↵
) CX UNLIMIT 4 ↵
```

This sequence tells users about the impending limit. (The BROADCAST macro is described in the *Installing* manual for your particular operating system. The LIMIT commands impose the CPU time limit of one hour on streams 1, 2, and 3 of the BATCH_INPUT queue and no limit on stream 4.

```
) BROADCAST Laser printer (QUEUE=LQP) limited to 20 pages. & ↵
&) For long files, use line printer (default queue) ↵

) CX LIMIT @CON27 20 ↵
```

Here the operator does not want users tying up the laser printer (device @CON27 running on queue LQP), so he or she limits it to 20 pages and sends a message to tell users about the limit and remind them that they can use the line printer (device @LPB running on the default queue, LPT) for long jobs.

LOGGING

Directs EXEC status and critical messages.

Format

CX LOGGING [*/START[/MAX=blocks] [pathname]* | */STOP* | */CONSOLE=@CONn* | */NOCONSOLE*]

where

/START	tells EXEC to begin logging all of its status messages to the log file named in pathname.
/MAX=blocks	is the maximum number of 512–byte disk blocks you want the log file to be. This value must be between 1 and 32,767.
pathname	is the pathname of the file to receive EXEC status messages. If you enter a filename only, EXEC creates the file in the directory @ (:PER) and uses it as the log file. If the file already exists, EXEC appends to it.
/STOP	tells EXEC to end logging its status messages.
/CONSOLE=@CONn	tells EXEC to send status messages to the terminal specified, instead of to the system console (@CON0 or @PMAP0). This switch doesn't affect the name or existence of the EXEC log file. Someone must be logged on the terminal specified.
/NOCONSOLE	tells EXEC not to send messages to a terminal previously specified with this command.

Description

If you omit arguments, EXEC tells you whether or not logging is on, and if so, the pathname of the logging file and the pathname of the terminal.

If you use the /START switch, EXEC starts logging all of its status messages to the file indicated by pathname. The optional /MAX=n switch allows you to limit the size of the logging file. If the file exceeds the maximum size limit, EXEC goes back to the beginning of the file and overwrites the information with current log messages. If you omit /MAX=, EXEC doesn't limit the file's size.

The /STOP switch tells EXEC to stop logging messages; it turns off logging to a file. The START and STOP switches do not affect logging to a terminal.

You can write text strings to EXEC's log file with the CX MESSAGE command.

By default, EXEC sends all status messages to the system console (@CON0 or @PMAP0). With the LOGGING command, you can redirect status messages, perhaps to your terminal, to a log file, or to both. You may decide that you don't want interruptions at your terminal (or at the system console, or any other terminal you designate), but still want to review errors. In this case, enter the /START and /NOCONSOLE switches. Don't worry about not receiving critical messages — EXEC still sends critical messages to the system console, even if you use the /NOCONSOLE switch. However, EXEC does not write critical messages to a file unless you specify logging and include a filename.

In addition to the LOGGING command, consider using the CLI CHARACTERISTICS command with the /NRM switch if anyone uses the system console to do work. The /NRM (no receive messages) switch prevents noncritical message interruptions from a variety of programs (such as XODIAC and CEO) that send messages to @CON0 or @PMAP0.

Why Use It?

An EXEC log can help you keep track of batch and print queue usage and user MOUNT requests. (Note that EXEC's log file is different from the operating system SYSLOG log file. SYSLOG generally is more useful for user accounting information. EXEC puts account–oriented information into SYSLOG, not the EXEC log. SYSLOG is described in Chapter 11, "Using Other Runtime Tools.")

Example

```
) CX LOGGING/START :UTIL:EXEC_LOG_NOV.93 ↵

.
.
.

) CX LOGGING/STOP ↵

) QPRINT EXEC_LOG_NOV.93 ↵
```

LPP

Sets the number of lines per page (LPP) for a device.

Format

CX LPP @printername n

where

@printername is the pathname of a printing device in :PER.

n is the number of lines per page. This number must be an integer between 6 and 144.

Description

The LPP command sets a new number of lines per page for this device. It overrides any previous VSGEN, CHARACTERISTICS/LPP, or EXEC LPP settings. The new LPP setting remains until you change it with EXEC's LPP command or bring down EXEC's XLPT cooperative process.

To change an LPP setting:

- Type CX PAUSE, the printer name, and press NEW LINE.
- When EXEC says the printer is paused, type the desired CX LPP command.
- Type CX CONTINUE, the printer name, and press NEW LINE.

The LPP command does nothing on a device where an EXEC DEFAULTFORMS or FORMS command is in effect.

Why Use It?

The default number of lines per page is 66. You may want to specify fewer lines; perhaps to produce more white space for greater readability. You may want to print a special form in the printer (although you'd usually use EXEC's FORMS or DEFAULTFORMS command for this).

Example

```
) CX PAUSE @LPB1 ↵
From EXEC time

) CX LPP @LPB1 60 ↵
) CX CONTINUE @LPB1↵
From Pid 3 : (EXEC) @LPB1 [Idle]
)
```

MAPPER

Sets character mapping at an EXEC printer.

Format

CX MAPPER @printername *[mapperfile-name]*

where

@printername	is the pathname of the printer in :PER; for example, @LPB.
mapperfile-name	is the name of a mapper file in the directory :UTIL:FORMS which defines the character mapping to be performed.

Description

Tells the XLPT process to use character mapping (translation) as described in the mapper file on the print device specified. The new map overrides any currently in effect for the device (set with EXEC's START command or the MAPPER command). The printer must be paused and idle before you issue the CX MAPPER command.

XLPT allows for the use of either ASCII 7-bit or the ASCII 8-bit Data General International (DGI) character set. Mapping allows a representation of the DGI character set to be printed on ordinary 7-bit printers.

A mapper file contains a series of commands, one per line. All lines must end with a line terminator (NEW LINE or Form Feed), including the last line. Unlike CLI command lines, mapper file command lines cannot be continued by typing an ampersand (&) prior to the line terminator. XLPT mapper files are discussed in detail in Appendix B, "XLPT Mapper Files," at the end of this manual.

Why Use It?

A mapper file is used when a file to be printed contains characters which are defined differently from their normal ASCII equivalents. For example, sometimes you may need to print documents that require using the Data General International (DGI) character set. The MAPPER command allows the DGI character set to be printed on 7-bit (as well as 8-bit) ASCII printers.

Example

```
) CX PAUSE @LPB1 ↵
From Pid 3 : (EXEC) @LPB1 [Idle]

) CX MAPPER @LPB1 DGI_CHARACTER_SET ↵
) CX START @LPB1 ↵
)
```

MDUMP

Creates a dumpfile of the EXEC process's memory image.

Format

CX MDUMP

Description

Only a user with the same username as EXEC's father (usually OP) or with System Manager privilege can issue this command. There are no arguments to the MDUMP command. When you issue this command, EXEC displays the following message:

(EXEC) taking a memory dump

The command blocks the EXEC process, so EXEC will not be able to respond to user requests until the command finishes. When the MDUMP command finishes, EXEC displays the message:

(EXEC) EXEC memory dump is

and adds the pathname of the dump file, using the format:

<exec_dir>:?EXEC.dd_mon_yy.hh_mm_ss.7.MDM

where

<exec_dir>	is the name of the directory containing EXEC.PR (usually :UTIL),
dd_mon_yy	are the day, month, and year,
hh_mm_ss	are the hour, minute, and second,
7	identifies the memory ring being dumped, and
MDM	is a standard filename suffix for a dump file.

The dump file has EXEC's default ACL, usually OP,OWARE.

If the MDUMP command fails, EXEC displays the message

(EXEC) memory dump failed

with the text of the error.

Why Use It?

Sometimes, you may want to take a dump of EXEC (for example, to accompany a Software Trouble Report, or STR). The MDUMP command allows you to create the memory dump.

Example

) SUPERUSER ON ↵

Su) CX MDUMP ↵

(EXEC) taking a memory dump

... (EXEC pauses as the dump occurs) ...

(EXEC) EXEC memory dump is

:UTIL:?EXEC.15_SEP_93.17_03_00.7.MDM

MESSAGE

Sends a message to EXEC's log file.

Format

CX MESSAGE message

where

message is a text string to be entered in EXEC's logging file.

Description

If EXEC is currently logging, the MESSAGE command sends the message to EXEC's log file. EXEC also echoes the message on the EXEC status terminal.

The maximum length of the MESSAGE command is 80 characters. However, you can type additional message commands if you want to say more.

If EXEC is not logging messages, it will not record the message in a log file, nor will it report an error.

Why Use It?

If EXEC is logging messages, you may want to put text messages in its log file for the record.

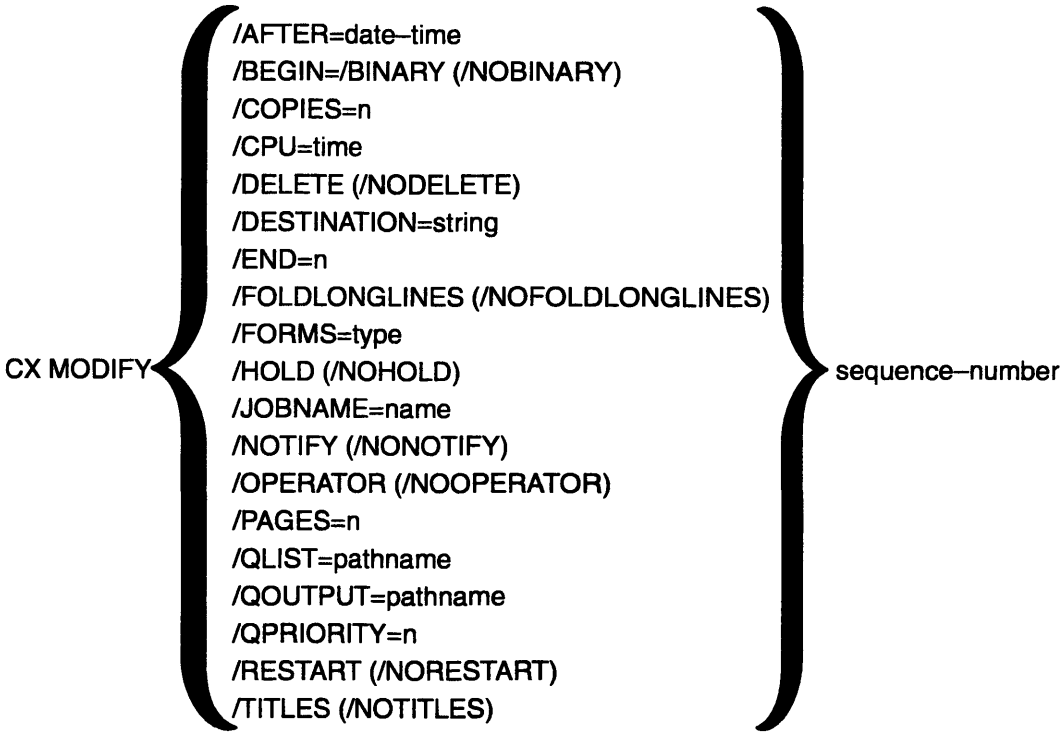
Example

```
) CX MESSAGE Running 4 streams 23-JUL-93 10am.
From Pid 3: (EXEC) Running 4 streams 23-JUL-93 10am.
)
```

MODIFY

Modifies existing inactive queue entries.

Format



where

/AFTER=date–time	processes the request after the date and time (dd–mon–yy:hh:mm:ss) or time (hh:mm:ss) specified. You can use a plus sign (+) followed by a time (hh:mm:ss) later in the day. Seconds and minutes are optional.
/BEGIN=n	starts printing the file at page n. The default value of n is 1.
/BINARY	prints in binary mode (only valid for devices with binary mode enabled).
(/NOBINARY)	doesn't print this file in binary mode.
/COPIES=n	produces n copies of the file. The default value is 1.
/CPU=time	limits the CPU time for jobs to the maximum time specified, in the form hours:minutes:seconds. (Specify an even number for seconds; otherwise the system subtracts one second.) The default is 1 minute.
/DELETE	deletes the pathname(s) after processing.
(/NODELETE)	retains the pathnames after processing.
/DESTINATION=string	prints the specified string at the top of any header or trailer pages. The default is username.

MODIFY (Continued)

/END=n	stops printing the file at page n. The default limit is the last page of the file.
/FOLDLONGLINES	continues long lines on the next line of the listing.
(/NOFOLDLONGLINES)	truncates long lines.
/FORMS=type	prints on the special form indicated.
/HOLD	holds the entry until explicitly released with the CLI QUNHOLD command.
(/NOHOLD)	releases the entry when finished.
/JOBNAME=name	names the entry as specified. You can use this name later with the QHOLD, QUNHOLD, or QCANCEL commands.
/NOTIFY	tells EXEC to send a message to your terminal upon completion of request.
(/NONOTIFY)	tells EXEC not to notify you of request completion.
/OPERATOR	indicates not to run this job unless an operator is on duty.
(/NOOPERATOR)	runs the job regardless of operator presence.
/PAGES=n	specifies the number of pages printing.
/QLIST=pathname	sets the generic list file of the batch process to the pathname indicated.
/QOUTPUT=pathname	sets the generic output file of the batch process to the pathname indicated.
/QPRIORITY=n	sets the priority of this job, from 0 to 255 if a value is specified, or to the value specified by the user's profile.
/RESTART	restarts the system if the system fails while processing this entry, starts processing the entry from the beginning when the system comes back up.
(/NORESTART)	doesn't begin processing again when the system comes back up again.
/TITLES	prints each page with a title line consisting of the file's pathname, date and time last modified, and page number.
(/NOTITLES)	tells EXEC not to print title lines on each page.
sequence–number	specifies the entry you're modifying.

NOTE: The command line must include at least one switch and can contain multiple switches.

Description

The MODIFY command lets you change the parameters of an inactive queued job. Unlike the CLI QMODIFY command, which lets users modify parameters of jobs that they own, the EXEC MODIFY command lets a user with the same name as EXEC (usually OP) modify the parameters of any inactive job. However, even the EXEC MODIFY command is invalid for any active job.

Table 3–10 shows which switches apply to which queue types.

Table 3–10 MODIFY Command Switches

Switch	Applicable Queue Type				
	Print	Batch	Networking	Mount	User
AFTER	X	X	X	X	X
BEGIN	X				
BINARY–NOBINARY	X				
COPIES	X				
CPU		X			
DELETE–NODELETE	X	X	X	X	X
DESTINATION	X				
END	X				
FOLDLONGLINES–	X				
NOFOLDLONGLINES	X				
FORMS	X				
HOLD–NOHOLD	X	X	X	X	X
JOBNAME		X	X	X	X
NOTIFY–NONOTIFY	X	X	X	X	X
OPERATOR–	X	X	X	X	X
NOOPERATOR	X	X	X	X	X
PAGES	X				
QLIST		X			
QOUTPUT		X			
QPRIORITY	X	X	X	X	X
RESTART–NORESTART	X	X	X	X	X
TITLES–NOTITLES	X				

MODIFY (Continued)

Why Use It?

You might notice that a user submitted a job with a misspelled forms filename. If the request were inactive, you could use the MODIFY command with the /FORMS= switch to correct the forms filename. Perhaps you've divided the queues on your system according to low and high priority jobs. You see that someone has submitted a low priority job in a high priority queue. With the MODIFY command, you could use the /QPRIORITY= switch to change the priority of the request (if inactive) to a value appropriate for that queue.

Example

The following command causes batch job number 11243 to stop processing after one minute of CPU time has been used. (The CPU= switch is valid only for batch entries.)

```
) CX MODIFY/CPU=1:00 11243 ↵
```

The following command changes a printing job so that two copies of the file will be printed.

```
) CX MODIFY/COPIES=2 576 ↵
```

MOUNTED

Tells EXEC that a tape is physically on a unit and on line.

Format

CX MOUNTED *[/MID=mid] [@tapeunit] [volid]*

where

/MID=mid	allows you to select any job in the list of mount requests. mid is an integer — the mount ID (sequence number) of the desired request.
@tapeunit	is the magnetic tape unit pathname in :PER; for example, @MTB1.
volid	is the volume identifier of an existing tape in the mount request.

Description

The MOUNTED command tells EXEC that you have physically placed a tape on a specific tape unit, and that the unit is on line. You must type CX MOUNTED and press NEW LINE, when prompted, to allow the user to access the tape.

Normally, EXEC processes mount requests based on the request's sequence number. If you want EXEC to service the next request, put the tape on the unit and type

) CX MOUNTED devicename ↵

If this is the second or subsequent volume of a request and you are using the same unit, you need type only CX MOUNTED and press NEW LINE.

However, if you are using more than one tape unit and have multiple requests outstanding, you must specify the mount ID of the desired request with the /MID switch, like this

/MID=mid

where mid is the mount ID of the request you want.

The MOUNTED command has EXEC create a link name in the user's initial working directory, to the tape unit (unlabeled tape) or file @LMT:volid (labeled tape). EXEC sets the unit's ACL to username,WARE or username,RE (depending on whether the user said /READONLY on his or her MOUNT command). This ACL prevents anyone else from using the unit.

The unit(s) assigned via the MOUNTED command belong to the user until you issue a DISMOUNTED command. Usually EXEC will prompt you for the DISMOUNTED command.

MOUNTED (Continued)

If OPERATOR is on (see OPERATOR command), EXEC will prompt you to type the MOUNTED command:

- whenever a user types (or batch job submits) an error–free CLI MOUNT command;
- when the system is ready for the next tape in a multi–volume request and you haven't premounted this tape;
- if you mount the wrong volume in a labeled tape request;
- when a user has made an implicit mount request (for example, using LOAD_II/V @LMT:FOO:SOURCES).

If OPERATOR is *not* on, the user's mount job will still be queued, but no operator will be notified of the mount request, and the mount job will receive no response.

With EXEC's *Unit Mount* prompt, there may be a *Request is* line with additional information from the user. The *Request is* line will not appear on an implicit mount request.

Why Use It?

EXEC's MOUNTED command and users' CLI MOUNT commands facilitate tape handling in a multiuser system where the mounting and dismounting of users' tapes is delegated, rather than handled by the users themselves.

Example

For an unlabeled mount, ROBIN types a CLI MOUNT command without a volid switch. The operator's terminal displays

```
From Pid 257 : (XMNT) 23–Jul–1993        13:30:00
                      **************************
                      Unlabeled Mount Request
                      **************************
                      MID = 81           USER = SCOTT
                      User Pid = 49      Requestor Pid = 49
                      Settings:          Default Density

                      Respond:           CX MOUNTED @[unitname]
                                or       CX REFUSED
```

EXEC repeats this message at intervals until you respond. You mount a tape on MTB0 and type

```
) CX MOUNTED @MTB0 ↵
```

and the user can use the tape.

A labeled tape user asks for a multi–volume labeled tape mount, knowing which volids to specify. EXEC prompts you

From Pid 257 : (XMNT) 23–July–1993 13:30:00
****
Labeled Mount Request
****
MID = 81 USER = PHIL
User Pid = 49 Requestor Pid = 73
Volumes: VOL0, VOL1, VOL2
Mount Volume: VOL0
Settings: Default Density

Respond: CX MOUNTED @unitname
or CX REFUSED

Let's say you mount the tapes on MTB0, MTB1, and MTB2 and decide to use the PREMOUNT command. You type

```
) CX MOUNTED @MTB0 ↵

) CX PREMOUNTED @MTB1 DB0034 COBOL ↵

) CX PREMOUNTED @MTB2 DB0035 COBOL ↵
```

The tape I/O will proceed. When it is done, the user will type the DISMOUNT command and EXEC will prompt you to dismount the tapes. There are other MOUNT examples in the section "Managing Mount Processing," earlier in this chapter.

MOUNTSTATUS

Displays status of each MOUNT and DISMOUNT request.

Format

CX MOUNTSTATUS *[mount–ID]*

where

mount–ID is a mount identifier, the integer number of a mount request in the mount queue.

Description

The MOUNTSTATUS command displays all mount or dismount requests or displays the mount request with the specified mount identifier (mount–ID).

If you omit the argument, EXEC reports each entry according to mount number. If there are no requests, EXEC reports *No mount requests*.

Why Use It?

MOUNTSTATUS is the most convenient way to check existing mount requests. You'll be using it often. (To check the status of tape units, use EXEC's UNITSTATUS command.)

Example

) CX MOUNTSTATUS ↵

Warning: No mount requests

Later on, MOUNSTATUS reveals that there are two mount requests:

) CX MOUNTSTATUS ↵

From Pid 257 : (XMNT) 23–Jul–1993 10:38:17

****

Unit Dismount Request

****

MID = 67	*USER = SCOTT*
User Pid = 47	*Requestor Pid = 63*
Volumes:	*VOL0, VOL1*
Units:	*MTB0/VOL0, MTB1/VOL1*
Dismount Unit:	*MTB0, MTB1*
User Message:	*Please mark the tapes as mine.*
Respond:	*CX DISMOUNTED @[unitname]*

From Pid 257 : (XMNT) 23–Jul–1993 15:07:44

```
**********************
Labeled Mount Request
**********************
MID =    82          USER = SCOTT
User Pid= 64         Requestor Pid= 59
Volumes:             VOL0, VOL1
Mount Volume:        VOL0
Settings:            1600 BPI, IBM Format, Write–Protect
User Message:        thank you!

Respond:             CX MOUNTED @unitname
          or         CX REFUSED
```

Here one tape request is awaiting a dismount and another is awaiting a mount. EXEC repeats these messages at intervals even if you don't use the MOUNTSTATUS command.

OPEN

Opens a queue to users (opposite of CLOSE).

Format

CX OPEN queuename

where

queuename is an existing queue you want to open.

Description

The OPEN command opens a queue so that users may submit requests to it.

Users access a queue via CLI commands like QSUBMIT, QBATCH, QPRINT, QPLOT,QDISPLAY, or QFTA. User programs can also access the queue by name via program OPEN, WRITE, and READ statements.

The first time EXEC is brought up, someone must open EXEC's BATCH_INPUT, BATCH_OUTPUT, and BATCH_LIST queues; someone must create and open the print and possibly networking queues. Once these queues are opened, they normally do not need to be opened again — even when the system is brought down and up again.

The CLI command QDISPLAY describes queues and their open and closed status.

Why Use It?

Each queue must be opened before it can be accessed; this is usually done the first time EXEC is brought up. Each time you create a new queue you must open it.

Also, if you close a queue for any reason, you must open it when you want it to accept requests.

Example

) CX OPEN LQP ↵

) CX START LQP @CON25 ↵

) CX CONTINUE @CON25 ↵

This sequence opens, starts, and continues the letter–quality printer queue, LQP, on the printer attached to line @CON25.

OPERATOR

Tells EXEC that an operator is or is not available to handle MOUNT requests.

Format

CX OPERATOR { ON *[@CONn]* / OFF }

@CONn is the name of the terminal to which EXEC sends messages, where n is the number of the terminal. If you omit @CONn, but specify CX OPERATOR ON, EXEC turns the OPERATOR command on at the system console. Typing CX OPERATOR OFF turns operator off at the console where it is currently on.

Description

An operator must be available (ON) for EXEC to process user mount requests. If an operator is not on duty (OFF), EXEC holds the mount requests in the queue until an operator is available (i.e., until he or she types CX OPERATOR ON and presses NEW LINE).

When EXEC is brought up, the operator status is off.

You (and any user) can tell whether the OPERATOR status is on or off with the !OPERATOR pseudomacro. For example,

```
) WRITE An operator is [!OPERATOR] duty. ↵
An operator is OFF duty.
)
```

If a privileged user issues the OPERATOR ON command while you are still an active operator, that person becomes the current operator. You will receive a message from EXEC stating that someone else has taken over operator responsibilities.

EXEC's OPERATOR command has no relation to the CLI command OPERATOR. The latter command enables the CLI to write and read labeled diskettes, described in Chapter 5.

OPERATOR (Continued)

Why Use It?

Some EXEC–controlled functions require a person in the role of system operator. For example, if a user requests a tape mount, someone must be available to mount the tape. The OPERATOR command tells EXEC that such a person is available.

If your system is to process user tape mount requests, you must issue the CX OPERATOR ON command and press NEW LINE. Later, if you leave your terminal and no one is available to handle such requests, issue the CX OPERATOR OFF command and press NEW LINE so that user requests will not wait in vain for operator service.

Example

) WRITE [!OPERATOR] ↵

OFF

) CX OPERATOR ON @CON25 ↵

From Pid 3 : (EXEC) System Operator On Duty at @CON25

This command makes @CON25 the current operator terminal until the EXEC OPERATOR command is issued with a different terminal name, OPERATOR is turned off, or the system comes down.

PAUSE

Tells stream(s) or a device to stop processing jobs (does not affect currently active job).

Format

CX PAUSE *[devicename] [n]*

where

devicename is the name of the device that you want to pause. If the device name is omitted, the default is BATCH_INPUT.

n indicates the number of the stream that you're pausing. If omitted, EXEC stops all streams associated with the device name specified.

Description

The PAUSE command stops processing at the specified stream or device. If a job is active there, it will complete before the pause takes effect. All incoming requests remain in the queue until you issue EXEC's CONTINUE command.

If you omit arguments, the PAUSE command pauses all BATCH_INPUT streams. To pause a different specific stream or device, give its name as an argument. Issuing a PAUSE command to a paused device or stream has no effect.

Why Use It?

You will want to stop job processing; for example, to suspend output before changing parameters (like priority) for a stream or device, or before shutting down EXEC.

Example

To pause all BATCH_INPUT streams you'd type

```
) CX PAUSE ↵
```

To pause the main line printer, @LPB, type

```
) CX PAUSE @LPB ↵
```

PREMOUNT

Tells EXEC you have mounted a volume before there was a specific request for it.

Format

CX PREMOUNT*[/IBM] [/DENSITY=value]* @tapeunit volid username

where

/IBM	is the switch indicating that you want the volid argument converted from ASCII to EBCDIC before EXEC compares it to the volid on the tape.
/DENSITY=value	tells the system to check the volid at the specified density and to permit only MOUNT requests that specify this density. *value* can be 800, 1600, or 6250 (bits per inch, if allowed by the unit); ADM (Automatic Density Matching); or LOW, MEDIUM (reverts to LOW on a dual–density drive), or HIGH. If you intend to use a tape on another unit, be careful when using LOW, MEDIUM, or HIGH: "low" or "high" on one unit may not be compatible with the densities on another unit, which would prevent loading from the tape. Omitting this switch tells the system to use the drive's default (specified in VSGEN) and to match this premount request with only those mount requests that do not specify a density.
@tapeunit	is the magnetic tape unit, preceded by @; for example, @MTB2.
volid	is the volume id of the tape you're premounting.
username	is the name of the user you are premounting the tape for.

Description

The PREMOUNT command forms an association between a volume and a tape unit on behalf of a user.

When you premount a volume, EXEC sets the ACL for that unit to null — meaning that no one but a superuser can use it. Then when the user's process needs that volume, EXEC changes the ACL to username,WARE or username,RE (depending on whether the user requested /READONLY on the mount). Later, when you dismount the tape, EXEC restores the ACL to its setting before the premount.

If you premount a volume and then change your mind before I/O to it occurs, use CX DISMOUNTED and the @tapeunit name to sever the connection set by premount. Then dismount the tape.

You cannot premount a volume for implicit mount requests. You must use the MOUNTED command for these.

PREMOUNT (Continued)

Why Use It?

The PREMOUNT command can be a big timesaver if you have more than one tape unit. When a user requests a multi–volume mount, you can mount the first volume on a unit; then use PREMOUNT for the second and subsequent volumes on other units.

The user process can then access all tapes as needed without operator intervention — even if an operator is not physically present. If you didn't use the PREMOUNT command, you'd have to mount each tape sequentially, as prompted by EXEC.

Example

The following command tells EXEC that you've premounted tape volume DB1103 on unit MTB10 for user SCOTT:

```
) CX PREMOUNT @MTB10 DB1103 SCOTT ↵
```

There is another PREMOUNT example in the MOUNTED command description.

PRIORITY

Changes the system priority and process type for streams and devices.

Format

CX PRIORITY [*/PREEMPTIBLE* | */RESIDENT* | */SWAPPABLE*] *[devicename] [n]*priority

where

/PREEMPTIBLE	makes the process type pre–emptible.
/RESIDENT	makes the process type resident.
/SWAPPABLE	makes the process type swappable. This is the default.
devicename	is the name of the device whose priority and/or process type you're setting. If the device name argument is omitted, the default is BATCH_INPUT.
n	indicates the number of the stream whose priority and/or process type you're setting. You must include this argument if you don't specify a device name.
priority	is an integer specifying the priority of the process. The range is 1 through 511 for any process type. 1 is highest, 511 lowest.

Description

The PRIORITY command changes the system priority for streams and for cooperative processes like XLPT. The default priority for printers and batch streams is 2.

The range of possible priorities, from highest to lowest, is 1–511. The range is divided in VSGEN into three groups — high, medium, and low (default limits are 1–255, 256–258, and 259–511). To change a process group, specify a priority that's unique in its group range. For example, priority 4 puts a process in group 1, and priority 259 puts it in group 3, assuming the default groups were chosen at VSGEN.

To change the process type, include the switch /PREEMPTIBLE or /RESIDENT. /SWAPPABLE is the default for streams and device processes.

To check the current process type and system priority, use CX STATUS for streams, or use CX SPOOLSTATUS for device processes.

"Processes and CPU Time" in Chapter 13 explains process and priority concepts.

Why Use It?

Sometimes, you may want to favor printing requests over other processing; if so, give a line printer a high priority, like 1 (or 4, to make it a group 1 process). To favor printing even more, make the process pre–emptible or even resident. The same principle applies to streams — you can give certain streams higher priority and/or make them pre–emptible or resident. (If you give many processes a high priority — or make them pre–emptible or resident — system performance may suffer.)

If you settle on specific nondefault process type and priority values, put the EXEC commands for them in the UP.CLI macro.

EXEC's PRIORITY command combines the functions of the CLI commands PRIORITY and PRTYPE, and it is tailored for EXEC operations. Any CLI process with the username of OP can issue it.

Example

```
) CX STATUS ↵

... (status information for the BATCH_INPUT queue, including type and priority) ...

) CX PRIORITY/RES 1 1 ↵

) CX PRIORITY 4 255 ↵

) CX SPOOLSTATUS @LPB ↵

... (status information, including type and priority) ...

) CX PRIORITY/PREEMPTIBLE @LPB 4 ↵

)
```

These commands check stream status with CX STATUS; then make stream 1 resident with a priority of 1 and give stream 4 the lowest priority, 255. Other streams retain the default types and values. Then the operator checks the status of @LPB and makes it pre–emptible with a priority of 4.

PROMPTS

Turns EXEC's time prompt off or on.

Format

CX PROMPTS { ON / OFF }

Description

If you set PROMPTS to OFF, only the CLI prompt appears when EXEC is ready for a command. If PROMPTS is on (default), EXEC displays the prompt

From EXEC hours:minutes:seconds

after each command.

Why Use It?

Sometimes EXEC's time–of–day prompt can be distracting, especially on a hardcopy terminal. On the other hand, you may like the prompt as an acknowledgement of your commands.

Example

```
) CX PROMPTS ON ↵

From EXEC 15:33:30
)
```

EXEC displays the time of day in hours (24–hour clock), minutes, and seconds.

PURGE

Deletes all inactive entries in a closed queue.

Format

CX PURGE queuename

where

queuename is the queue you want to purge.

Description

The PURGE command deletes all inactive entries in the specified queue. After the command succeeds, the queue is empty. The queue should be closed before the queue is purged. Use EXEC's CLOSE command to do this.

Why Use It?

Sometimes, you'll want to clean out a queue; for example, if it has a lot of useless requests or if you want to delete the queue. Use PURGE to delete all inactive requests, and CANCEL for an active one.

It is not always necessary to close a queue in order to service a device such as a printer. You might want to use the CX PAUSE command to pause the queue. This stops requests from being printed until the device is repaired, but still allows users to submit requests. Requests submitted while the queue is paused will be printed when the device is continued later.

To get rid of an individual request in a queue, use EXEC's CANCEL or FLUSH command instead of the PURGE command.

PURGE (Continued)

Example

To purge a queue named LPT:

```
) CX PAUSE @LPB ↵
) CX CLOSE LPT ↵
) CX PURGE LPT ↵
) CX FLUSH @LPB ↵
) CX OPEN LPT ↵
) CX CONTINUE @LPB ↵
)
```

Note that on a slow printer such as a letter–quality printer, it can take a moment for the buffer to empty and the printer to stop.

QPRIORITY

Displays or changes the qpriority range to be accepted by a device or its stream(s).

Format

CX QPRIORITY *[devicename] [n high–value low–value]*

where

devicename is the name of the device whose queuing priority range you're displaying or setting. If the device name argument is omitted, the default is BATCH_INPUT.

n indicates the number of the stream whose queuing priority range you want to display or set.

high–value is the highest priority value of a request that the stream or device can accept. It must be an integer between 0 (highest) and 255 (lowest).

low–value is the lowest priority value of a request that the stream or device can accept. It must be an integer between 0 and 255.

Description

The QPRIORITY command displays or changes the range of priority that a specified stream or device will process.

If you omit arguments, EXEC displays the queue priority for all BATCH_INPUT streams. If you specify a different stream number or device name, EXEC displays its queue priority.

Users specify queue priority with the /QPRIORITY switch when submitting a queue request. If a user omits a /QPRIORITY= switch when submitting a request, EXEC assigns the request a queue priority. This is the median of the highest queue priority and the lowest queue priority indicated by the user's profile. For example, if a user's highest queue priority is 0 (PREDITOR default), then the user's jobs are assigned a queue priority of 127, which is halfway between 0 and 255.

Do not confuse QPRIORITY with process priority, set with the CLI command PROCESS or PRIORITY, or EXEC's command PRIORITY.

QPRIORITY (Continued)

Why Use It?

Your installation might want to have requests processed in different queues, depending on each request's priority. For example, you could establish different queue priority ranges for two streams, and give users some control over queue processing. Users may request a certain queue priority with the /QPRIORITY= switch.

If a user's MAX QPRIORITY, set in the profile during the PREDITOR dialog, is less than other users' priorities (that is, farther from 0), his or her printing requests will have less priority than those of other users, regardless of the EXEC QPRIORITY command.

Example

) CX QPRIORITY ↵

... (EXEC displays current QPRIORITY for BATCH_INPUT streams) ...

) CX QPRIORITY 1 0 100 ↵

) CX QPRIORITY (2,3) 101 255 ↵

After this sequence, requests submitted with QBATCH/QPRIORITY=0 through QBATCH/QPRIORITY=100 will go to stream 1. Requests submitted with QBATCH/QPRIORITY=101 through QBATCH/QPRIORITY=255 will go to streams 2 and 3. Simple QBATCH requests, which generally get priority 127, will also go to streams 2 and 3. Stream 4, if continued, will accept all requests.

To set different priorities for two line printers:

) CX QPRIORITY @LPB 0 126 ↵

) CX QPRIORITY @LPB1 127 255 ↵

After this sequence, print requests submitted with QPRINT/QPRIORITY=0 through QPRINT/QPRIORITY=126 will go to printer LPB, and requests submitted with QPRINT/QPRIORITY=127 through QPRINT/QPRIORITY=255 will go to printer LPB1.

REFUSED

Tells EXEC and a user that you refused a MOUNT request.

Format

CX REFUSED *[mount–ID]*

where

mount–ID is the number the system assigns the user's mount request.

Description

The REFUSED command tells XMNT to reject a user mount request (for whatever reason). It also sends the message *Refused by operator* to the user who typed the MOUNT command. You can omit the mount ID for the last (or only) request.

Using the mount–ID is most useful when you have multiple requests. EXEC deals with dismount requests on a first–come, first–served basis. Thus, EXEC is likely to dismount the first request it sees, which may not be the request you want dismounted. Using mount–ID ensures that EXEC acts on the proper request.

Why Use It?

There may be times when you want to reject a mount request. This can be your own mount request or another user's. For the latter, you can also use the SEND command to explain why you refused.

Example

From Pid 257 : (XMNT) 23–Jul–1993 13:30:00

```
**************************
Unlabeled Mount Request
**************************
MID =    81          USER = SCOTT
User Pid = 49        Requestor Pid = 49
Settings:            Default Density

Respond:             CX MOUNTED @[unitname]
          or         CX REFUSED
```

) CX REFUSED ↵

) SEND 44 Sorry Dave – backup time. Try again in 2 hours. Thanks. ↵

The following messages appear on Dave's terminal:

Warning: Refused by operator, MOUNT
From PID2: Sorry Dave – backup time. Try again in 2 hours. Thanks.

RELEASE

Removes a tape unit from EXEC's list of mountable units (opposite of ALLOCATE).

Format

CX RELEASE @tapeunit

where

@tapeunit is the pathname of a tape unit in :PER; for example, @MTB1.

Description

The RELEASE command tells XMNT to remove a tape unit from its list of mountable units, which prevents user tape mounts from occurring on the units until someone re–enables mounts with EXEC's ALLOCATE command (or until EXEC is halted and restarted). While a unit is released, its name is not displayed by the UNITSTATUS command.

Why Use It?

This command is useful when you want to prevent user tape mounts on a specific unit. Also, you can use the RELEASE command to force the UNITSTATUS command to display only units you have, instead of all units supported by each controller. Having only existing units displayed simplifies labeled tape operations.

Example

) CX UNITSTATUS ↵	UNITSTATUS displays all mountable units.
@MTB0 Not Mounted *@MTB1 Not Mounted* ... *@MTB7 Not Mounted*	
) CX RELEASE @MTB(1,2,3,4,5,6,7) ↵	Remove all units except 0 from mountable list.
) CX UNITSTATUS ↵	
@MTB0 Not Mounted	UNITSTATUS displays only unit 0.

RESTART

Restarts processing of the current job, optionally specifying a range of pages to print.

Format

CX RESTART *[devicename] [n] [start–page] [end–page]*

where

devicename is the name of a device; for example, @CON25. If no device is specified, the default is BATCH_INPUT.

n indicates the number of the stream whose active request you're restarting. This is optional. If you don't use it, you will restart all streams started for this device.

start–page is the first page of the file to be restarted; it must be an integer between 1 and 65535.

end–page is the last page of the file to be printed. It must be an integer, larger than the start–page, and be between 1 and 65535.

Description

The RESTART command restarts processing of the current job on the specified device. You can specify beginning and ending page numbers for printing. If you omit page numbers, EXEC's XLPT process uses the default start– and end–page numbers 1 and 65535 respectively. If you specify only one number, XLPT uses it as the start–page number and uses the default value as the end–page number.

If the user included a /NORESTART switch when submitting the request, the RESTART command will not work and an error message will be returned.

You can restart a job any time you want. With print jobs, the XLPT process always writes the current output buffer before restarting the file.

Why Use It?

Occasionally, a printer may jam or otherwise malfunction in the middle of a job. EXEC's RESTART command allows you to restart the job, optionally at a specific page. Printing specific pages can be very useful for printing on a slow device (like a letter–quality printer), or for parts of large printing jobs. You may find that a user forgot to move a needed object file into a directory prior to a link. You could move the .OB file into the directory and restart the job.

RESTART (Continued)

Example

To restart printer @LPB and print the entire current file:

```
) CX RESTART @LPB ↵
```

To restart a letter–quality printer and print pages 39 and 40:

```
) CX RESTART @CON25 39 40 ↵
```

To restart the batch request currently active in the fifth stream of the batch input queue named BATCH2:

```
) CX RESTART BATCH2 5 ↵
```

SILENCE

Suppresses EXEC messages about a stream (or streams) or device.

Format

CX SILENCE *[devicename] [n]*

where

devicename is the name of the device whose messages you're suppressing; for example, @LPB. The default is BATCH_INPUT.

n indicates the number of the stream that you're suppressing. If omitted, EXEC suppresses all messages about the streams associated with the device specified.

Description

The SILENCE command tells EXEC not to send messages to the system console (@CON0 or @PMAP0, by default) regarding a device or its streams. By default, EXEC prints brief messages, described under EXEC's BRIEF command.

If you omit the device name argument, EXEC suppresses messages about all BATCH_INPUT streams. If you specify a stream or device, EXEC suppresses messages from the source indicated. If you specify @CONSOLE, EXEC suppresses messages about terminal enabling and disabling.

The SILENCE command also suppresses messages to a different terminal, if specified with EXEC's LOGGING command, as well as to EXEC's logging file if logging is on. To restore message output, use EXEC's UNSILENCE command.

Why Use It?

You may not care about the status messages that EXEC sends. The SILENCE command ensures that status messages don't interrupt a user working at the system console (@CON0 or @PMAP0) or another terminal to which you've directed messages. However, with SILENCE you can't ever review these messages because you never see them. To get the best of both worlds, use the EXEC LOGGING command to tell EXEC not to display status messages on any terminal, but to record them in a file. Refer to the LOGGING command description earlier in this chapter.

In addition to the EXEC LOGGING and SILENCE commands, consider using the CLI CHARACTERISTICS command with the /NRM switch if anyone uses the system console to do work. The /NRM (no receive messages) switch prevents noncritical message interruptions from a variety of programs (such as XODIAC and CEO) that send messages to @CON0 or @PMAP0.

SILENCE (Continued)

Example

) CX SILENCE ↵

) CX SILENCE @LPB ↵

... (No EXEC batch messages from BATCH_INPUT or device messages to system console) ...

) CX UNSILENCE ↵

) CX UNSILENCE @LPB ↵

These commands silence all streams in BATCH_INPUT and printer @LPB; then the UNSILENCE command restores message output.

) CX SILENCE 4 ↵

This command silences the fourth stream of the BATCH_INPUT queue.

SPOOLSTATUS

Displays queue and device information.

Format

CX SPOOLSTATUS [*devicename* | *queuename*]

where

devicename is the name of a device; for example, @LPB.

queuename is a queue.

Description

The SPOOLSTATUS command describes the devices used by a queue, the queues using a device, or both. If you omit arguments, the command tells you

- each device name and the queue name(s) associated with it;
- the CPL, LPP, HEADERS, and TRAILERS settings;
- whether limiting is enabled and what the limits are; if not, the defaults;
- whether or not even pagination or binary mode is enabled;
- if a DEFAULTFORMS command is in effect;
- if a FORMS command is in effect;
- the priority range of the device as set by EXEC's PRIORITY command; and
- 7–bit/8–bit

If you use a queuename argument, the SPOOLSTATUS command tells you which device(s) the queue is associated with.

If you use a devicename argument, the SPOOLSTATUS command gives the information above for that device.

SPOOLSTATUS (Continued)

Why Use It?

The SPOOLSTATUS command is your primary status checking command for queues (as STATUS is for streams and devices and MOUNTSTATUS is for MOUNT and DISMOUNT requests). You will use it often, along with the CLI's QDISPLAY, to see what's happening with queues.

You'll probably use SPOOLSTATUS routinely before and after changing a device specification (as with CPL or FORMS).

If a user has been given read access to a device with the ACCESS command, he or she can use the SPOOLSTATUS command to get information about that device.

Example

```
) CX SPOOLSTATUS ↵

... (Information) ...

) CX SPOOLSTAT @LPB ↵

@LPB processing: BATCH_OUTPUT, BATCH_LIST, LPT,
CPL = 80, LPP = 66, Headers = 2, Trailers = 0
Even pagination enabled, Binary mode disabled
Bias factor = 0, Process type = Swappable, Priority = 2
```

START

Associates a queue with a device, assigning a cooperative process to manage the device (opposite of STOP).

Format

CX START *[/NL] [/NAME=process name]* [*/7BIT* / */8BIT*] queuename devicename *[file]*

where

/NL	tells the XLPT process to print each NEW LINE as carriage return NEW LINE, as needed on laser printers and some nonstandard printers.
/NAME=process name	assigns control of the device to the cooperative process with *process name*. If you supply a simple process name, and the process does not exist, EXEC creates it, running the default cooperative program for the queue's type. (You can supply a full process name only for existing cooperative processes.) If the device is already associated with a cooperative, EXEC ignores this switch. The default username portion of the process name is OP. If you omit this switch, EXEC uses the default process name for the queue's type — OP:*xxxxn*, where *xxxx* is XBAT, XLPT, XMNT, or XNET and *n* is a unique identifier.
/7BIT	tells the XLPT process to use 7 bits when it sends characters to the printer. This is usually the default.
/8BIT	tells the XLPT process to use all 8 bits when it sends characters to the printer. Without this switch, XLPT will send only 7 bits. If you have an 8–bit printer and want it to print special (8–bit) characters, you must include this switch. If the device is on a terminal line (like a letter–quality printer), the line must also have been configured for 8 bits (via the CLI command CHARACTERISTICS or via VSGEN — described in the *Installing* manual for your particular operating system.
queuename	is the name of the queue you want to start. The queue must already exist. It can be a permanent queue, like BATCH_OUTPUT; or it can be a queue created and opened via EXEC's CREATE and OPEN commands, such as an additional print queue.
devicename	is the name of a device; for example, @LPB1.
file	is the name of the mapper file containing keyboard characters and their conversions. The file must be in the directory :UTIL:FORMS.

START (Continued)

Description

The START command associates a queue with a device, assigning a cooperative process to manage the device. You must issue a START command for each device you want EXEC to handle. The device is usually a printer, but it can be a disk file or magnetic tape unit.

In addition, you can create a file containing infrequently used keyboard characters and the characters that you want printed instead of the character entered.

EXEC can manage more than one queue on a device. The UP.CLI macro makes use of this, by starting queues BATCH_OUTPUT, BATCH_LIST, and LPT on device @LPB. You can also start the same queue at multiple devices. For example,

```
) CX START LPT @LPB ↵

) CX START LPT @LPB1 ↵
```

associates the main print queue with two printers and will send print requests dynamically to alternate printers allowing the LPT queue to process two requests at once.

The /NL switch directs the XLPT process to convert each NEW LINE to a carriage return followed by a NEW LINE for printing. The additional carriage return is needed for any printer — such as many laser printers — that does not automatically position at the start of a line after printing a NEW LINE character. If carriage return is not added to NEW LINE on such a printer, the file may not print properly. In any case, the additional carriage return does no harm. To see if NEW LINE conversion is in force, use the SPOOLSTATUS command.

The /NAME switch lets you assign control to a specified process, as you would need to do with a cooperative process you have created. For more information about user cooperatives, see the file COOP_TOOLKIT.DOC in the directory COOP_TOOLKIT.

Using the mapper file UPPER turns on case conversion for printers. Without this filename as an argument, case conversion is not enabled and lowercase characters may not print at all on an uppercase–only printer. To start multiple queues on an uppercase–only printer, include the UPPER argument in the first START command; omit it from subsequent START commands to the printer. Include the UPPER argument for a line printer that has only uppercase characters.

Note that each printer involved must be on line when the START command is issued. If not, the XLPT cooperative process will not take control of the printer. You will need to put the printer on line and reissue the START command.

When you start a device — but before you continue it — you can issue device parameter commands like CPL, FORMS, and HEADERS.

After you start one or more queues on a device, you must continue the device (CONTINUE command) to allow it to process jobs.

Why Use It?

Each time the system comes up, someone must start each queue — except BATCH_INPUT and MOUNTQ — on a device before EXEC can process requests. Usually, the UP.CLI macro has commands to do this.

If a device fails or a cooperative terminates while the multiuser environment is running, you will need to use the START command to restart it. If a printer is down, you can pause the printer and stop the printer queue. Then you can start its queue on another device (perhaps a disk or tape file), which you can load and print on another machine.

The START command is also needed for other EXEC cooperative processes, like XODIAC network processes. It is not normally needed for batch input or mount processes, unless they have been explicitly stopped. (Or if they aborted; see the last example, below.)

Examples

The following commands are from an UP.CLI macro:

```
CONTROL @EXEC START (BATCH_<OUTPUT LIST> LPT) @LPB
CONTROL @EXEC CPL @LPB 85
CONTROL @EXEC CONTINUE @LPB
```

The first command line expands to start queues BATCH_OUTPUT, BATCH_LIST, and LPT on @LPB; the second command sets 85 characters per line; and the third continues the device. The first and third commands are the ones you should issue if the XLPT process somehow terminates abnormally.

To start a queue on a letter–quality printer:

```
) CX START LQP @CON15 ↵
) CX HEADERS @CON15 0 ↵
) CX BINARY @CON15 CLEANUP_FILE ↵
) CX CONTINUE @CON15 ↵
```

These commands start the letter–quality printer (CON15) as queue LQP, for CEO usage; then set it up and continue it. The CEO.PRINTER macro, shipped with the CEO system, issues these commands.

The following command tells EXEC to associate all requests sent to the FTQ queue with the cooperative process that owns the IPC port @FTA_EXEC.

```
) CX START FTQ @FTA_EXEC ↵
```

The next command tells EXEC to create an XLPT process for the LPB device, and to send requests from the LPT queue to that XLPT. The command also instructs EXEC to print output in uppercase using the mapper file UPPER in the directory :UTIL:FORMS.

```
) CX START LPT @LPB UPPER ↵
```

To restart the XMNT process (if it aborted, for example), type the following commands:

```
) CX START MOUNTQ MOUNTQ ↵
) CX OPERATOR ON @CON0 ↵
```

STATUS

Describes status of streams or devices.

Format

CX STATUS *[devicename] [n]*

where

devicename is the name of the device whose status you're displaying. The default is BATCH_INPUT.

n indicates the number of the stream whose status you're displaying. If omitted, EXEC displays the status for all streams of the device specified.

Description

The STATUS command tells you the active or paused status of the stream or device specified. If the stream or device is active, EXEC displays the

- sequence number;
- user's queue priority;
- username;
- process ID (PID);
- pathname of the source file;
- process type and process priority as defaulted or set by EXEC's PRIORITY command; and
- current page being processed and the number of copies left to print (active printers only).

If the device or stream is not processing a job, EXEC displays an *Idle* message; then describes the EXEC process type and priority.

If you omit arguments, EXEC displays the status of all the streams of BATCH_INPUT. If you specify a different device name, or a stream number, EXEC displays the status of that stream or device.

Why Use It?

The STATUS command is the primary status checking command for streams and devices. (To check queues, use EXEC's SPOOLSTATUS command.) You'll use the STATUS command often, to see what's happening with devices.

You'll probably use the STATUS command routinely before and after changing device parameters (as with QPRIORITY); you'll also use it to see what's going on when you plan to shut down EXEC.

The additional information given by STATUS can be handy when you want to align a device or see how long the current request will take.

If a user has been given read access to a device with the ACCESS command, he or she can use the STATUS command to get information about that device.

Example

```
) CX STATUS ↵

BATCH_INPUT_1 [Idle]
Bias factor = 0, Process type = Swappable, Priority = 2

BATCH_INPUT_2 [Idle]
Bias factor = 0, Process type = Swappable, Priority = 2

BATCH_INPUT_3 [Idle]
Bias factor = 0, Process type = Swappable, Priority = 2

BATCH_INPUT_4 Sequence number=3650 Qpriority=127 User = Roger
PID = 37, Pathname :UDD:DATA:?029.CLI.004.JOB, Maximum time = 36:24:30
Bias factor = 0, Process type = Swappable, Priority = 2
```

This is a status report on all BATCH_INPUT streams. Streams 1, 2 and 3 are idle; stream 4 is active.

```
) CX STATUS @LPB1 ↵

@LPB1, Sequence number = 143, Qpriority = 127, User = GABRIEL
PID = 6, Pathname :UTIL:PARU.32.SR
Current page = 9, Copies left = 1
```

This status report on @LPB1 shows the current page and copies left to print, after other things.

STOP

Dissociates a queue from a device (opposite of START), stops a device, or does both.

Format

CX STOP { queuename | devicename | queuename devicename }

where

queuename is the the name of a queue you want to dissociate from its device(s).

devicename is the name of a device; for example, @LPB1.

Description

The STOP command dissociates a queue from a device, stops a device, or does both — depending on the form you use.

- CX STOP queuename dissociates the queue from all associated devices. If this is the only queue associated with a device, the device stops and the pertinent EXEC cooperative process (for example, XLPT) terminates.
- CX STOP devicename dissociates the device from all associated queues and stops the device. The pertinent EXEC cooperative process terminates.
- CX STOP queuename devicename dissociates only the queue named from the device named. This is useful only if there are multiple queues associated with a device (or vice versa — multiple devices on a queue) and you want to sever only this queue from this device; otherwise you would use one of the forms above.

In all cases, the pertinent device finishes processing an active request. The queue accepts new requests and remains open. but the device won't process other requests in the queue (if any) until you issue the START command.

Why Use It?

You might use STOP when you want to stop a device and think about the current queue–device situation. You might use it if you want to get rid of a queue (but you must also close and perhaps purge the queue before you can delete it).

If things are okay and you want an orderly shutdown, use EXEC's PAUSE command instead of the STOP command. To stop an active request immediately, use EXEC's FLUSH command.

To prevent a queue from accepting new requests, use EXEC's CLOSE command.

Example

To dissociate queue LQP from its device:

```
) CX STOP LQP ↵
```

To dissociate all queues from a device and terminate EXEC's device–handling cooperative process:

```
) CX STOP @CON25 ↵
```

... (Finishes current request) ...

From Pid 3: (EXEC) @CON25 Cooperative terminated

To stop the BATCH_INPUT queue and device, and then continue BATCH_INPUT with a different number of streams:

```
) CX STOP BATCH_INPUT ↵
) CX START/STREAMS=10 BATCH_INPUT BATCH_INPUT ↵
) CX CONTINUE BATCH_INPUT ↵
```

Notice that the START command requires both a queuename and a devicename; thus, BATCH_INPUT appears twice, since BATCH_INPUT refers to both a queue and a device.

TERMINATE

Terminates the user process running on a terminal or immediately halts processing on a running device.

Format

CX TERMINATE *[/NAME=process name]* { devicename | @consolename }

where

/NAME= process name	is used when the process to be terminated is not a son of EXEC — i.e., is a special cooperative and not one of the standard processes created by EXEC. Give the full process name (pathname) of the process.
devicename	is the name of a device whose cooperative process you want to terminate. Note that the process will be terminated even if it is managing other devices in addition to this one. Use this form of termination with care.
@consolename	is the pathname of the terminal on which you want to terminate the process.

Description

The TERMINATE @consolename command terminates the user process on the specified terminal and logs the user off. The user's terminal displays the messages

Process n terminated BY OPERATOR COMMAND
Connect time hours:minutes:seconds
User user logged off @CONn date time

The TERMINATE devicename command terminates processing on all devices controlled by a cooperative. If that process is a son of EXEC, EXEC will terminate the cooperative process. Therefore, you can terminate cooperative processes like XLPT as well as terminal jobs. You cannot use EXEC's TERMINATE command to terminate a XODIAC file transfer agent (FTA) cooperative. Use the DOWN.NETWORK.CLI macro, supplied with XODIAC, to terminate XODIAC agent processes.

Why Use It?

A time may come when you must terminate a user process to protect other users or the system. You can use EXEC's TERMINATE command for this.

You may need to halt processing on a device immediately. Use the TERMINATE command in this circumstance; the START command can be used to resume processing for this device.

Example

This dialog ultimately terminates the user process on CON5.

) SEND 13 Log off NOW! ↵	Warn user.
... (a minute or so passes) ...	Wait.
) WHO 13 ↵	See if PID still exists.
PID 13 ABEL CON5 :CLI.PR	Still running on terminal CON5.
) TERM 13 ↵	Try to terminate it with CLI's TERMINATE command.
Warning ... Process not in hierarchy	Doesn't work.
) CX TERM @CON5 ↵	Try EXEC's TERMINATE command.
)	EXEC's TERMINATE works.

TRAILERS

Changes the number of trailer sheets after each printed file.

Format

CX TRAILERS @printername { 0 | 1 | 2 }

where

@printername is the pathname of a printer; for example, @LPB1.

0, 1, or 2 is the number of trailer sheets you want.

Description

The TRAILERS command sets the number of trailer sheets that follow the output for each printing request on the specified device. By default, EXEC provides no trailer sheet. Information on each trailer sheet includes

- destination name;
- username;
- queue name;
- device name;
- number of pages printed; and
- system identifier (SYSID).

EXEC's SPOOLSTATUS command describes the number of trailers on a device.

As with the HEADERS command, you must pause the device before EXEC will accept the command, and you must continue it afterward.

Why Use It?

Your organization might want one or more trailer sheets between each file for greater separation of printing jobs or to describe the number of pages printed. (The operating system command SYSLOG, described in Chapter 11, logs pages printed by each user.)

Example

To set the number of trailers on @LPB1 to 1, type

```
) CX PAUSE @LPB1↵
) CX TRAILERS @LPB1 1 ↵
) CX CONTINUE @LPB1↵
```

From Pid 3 : (EXEC) @LPB1 [Idle]

UNHOLD

Negates a previous HOLD command.

Format

CX UNHOLD { sequence–number | /USERNAME= }

where

sequence–number	is the sequence number of the request you want to release (unhold).
/USERNAME=	is the name of the user whose request(s) you want to release (unhold).

Description

The UNHOLD command cancels a previous EXEC HOLD command. It does not cancel a user's QHOLD or QPRINT/HOLD command — the user must use the QUNHOLD command for this. Use the CLI command QDISPLAY to check the hold status of a queue entry.

EXEC does not keep track of whether you held (HOLD command) a request by sequence number or username. So you can release any request from hold by using either its sequence number or its username. The sequence number releases only the request with the sequence number; the username releases all requests with that username.

See EXEC's HOLD command for more information.

Why Use It?

You must use UNHOLD if you want EXEC to process a request that you previously suspended with the HOLD command.

Example

To remove the previous operator hold on request 588:

```
) CX UNHOLD 588 ↵
```

UNITSTATUS

Displays the status of one or all tape units.

Format

CX UNITSTATUS *[@tapeunit]*

where

@tapeunit is the pathname of a magnetic tape unit; for example, @MTB12.

Description

Without an argument, the UNITSTATUS command displays each unit's status. With an argument, it describes a unit. If a unit is mounted, EXEC displays the following:

- the username;
- user process ID (PID);
- the volid(s) for a labeled tape request; and
- any premount on the unit.

Why Use It?

The UNITSTATUS command is better than the MOUNTSTATUS command if you are more interested in units than in mount requests. You might use it to decide which unit to use for a premount.

Example

) CX UNITSTATUS ↵

@MTB0 Not Mounted
@MTB2 Premounted User=ANDERS, VOLID=GP6423
@MTB1 Mounted MID=40, User=ANDERS, PID=9, VOLID=GP6422

UNLIMIT

Stops limiting on a queue or device (negates LIMIT).

Format

CX UNLIMIT *[devicename] [n]*

where

devicename is the name of the device for which you want to stop limiting. The default name is BATCH_INPUT.

n indicates the number of the stream for which you want to stop limiting.

Description

The UNLIMIT command negates a previous EXEC LIMIT command. If you omit arguments, EXEC removes the CPU time limit on all BATCH_INPUT streams. If you specify a device name but not a stream, EXEC removes the time limit or pages limit previously set for any streams for that device. If you use a stream argument, EXEC removes the time or pages limit for that stream.

See EXEC's LIMIT command for more information.

Why Use It?

If limiting is enabled, you must use the UNLIMIT command to lift the limits. (To set new nondefault time/page limits, use another EXEC LIMIT command.)

Example

... (Limiting has been enabled on streams) ...

... (Hours pass while streams run) ...

) BROADCAST Am unlimiting all streams in BATCH2.

) CX UNLIMIT BATCH2

)

... (Limiting has been enabled on printer @LPB) ...

) BROADCAST Am unlimiting main printer.

) CX UNLIMIT @LPB

UNSILENCE

Restores EXEC display of messages (negates SILENCE).

Format

CX UNSILENCE *[devicename] [n]*

where

devicename is the name of the device whose messages you're restoring. If the device name is omitted, the default is BATCH_INPUT.

n indicates the number of the stream whose messages you're restoring. If omitted, EXEC restores all messages from streams of the device specified.

Description

The UNSILENCE command negates a previous EXEC SILENCE command. Then each time it queues or processes a request, EXEC sends a message to @CON0 or @PMAP0 (by default) or another terminal specified with the EXEC LOGGING command. EXEC also records messages in its logging file if EXEC LOGGING is started. The contents of each message depends on whether an EXEC VERBOSE or BRIEF command is in effect. If you specify @CONSOLE, EXEC sends messages about enabling and disabling of terminals.

If you omit arguments, EXEC sends messages about all BATCH_INPUT streams. If you enter a different stream or device name argument, EXEC sends messages about the stream, or device.

Why Use It?

Having silenced EXEC, you might want to restore its batch or messages. However,after issuing the UNSILENCE command, you could use EXEC's LOGGING command to record messages in a log file without having them interrupt you or a user at a terminal. Refer to the LOGGING command described earlier in this chapter.

In addition to the EXEC LOGGING and SILENCE commands, consider using the CLI CHARACTERISTICS command with the /NRM switch if anyone uses the system console to do work. The /NRM (no receive messages) switch prevents the system console from receiving noncritical message interruptions from a variety of programs (such as XODIAC and CEO) that send messages to @CON0 or @PMAP0.

UNSILENCE (Continued)

Example

```
) CX SILENCE BATCH2 ↵

) CX SILENCE @LPB ↵

... (Hours of BATCH2 queue silence pass) ...

) CX UNSILENCE BATCH2 ↵

) CX UNSILENCE @LPB ↵
```

Here the operator silences EXEC messages from the BATCH2 queue; then restores them with the UNSILENCE command.

VERBOSE

Tells EXEC to give detailed messages (opposite of BRIEF).

Format

CX VERBOSE *[devicename] [n]*

where

devicename is the name of the device whose messages you're making more detailed (verbose). If the device name is omitted, the default queue is BATCH_INPUT.

n indicates the number of the stream for which you want more detailed messages. If omitted, EXEC makes all streams of the specified device verbose.

Description

When a stream or device accepts or processes a request, EXEC sends a message to the system console (@CON0, by default). This message may be either brief or verbose. The EXEC BRIEF and VERBOSE commands determine the verbosity of EXEC messages sent. Each BRIEF or VERBOSE command overrides the current message setting. BRIEF is the default setting.

VERBOSE messages include all BRIEF information, plus the pathname of the request's source file.

Why Use It?

There may be situations where you want to know the pathname of user request source files especially if EXEC logging is started. Use the VERBOSE command to have the pathnames logged. (If you use EXEC logging, you can direct messages to the log file specified rather than to a terminal.)

Example

From Pid 3 : (EXEC) BATCHQ, STRM=3 SEQ = 446 QPRI=127, USER = ROBIN

) CX VERBOSE BATCHQ ↵

From Pid 3 : (EXEC) BATCHQ, STRM=3 SEQ = 446 QPRI=127, USER = ROBIN
Pathname = :UDD:F77:NAVY_TEST:?016.CLI.002.JOB

Here you can see the difference between the BRIEF (default, at top) and VERBOSE messages.

XHELP

Describes EXEC commands.

Format

XHELP *[command]*

where

command is an EXEC command.

Description

If you omit command, XHELP lists all the EXEC commands. If you include an EXEC command, XHELP describes that command.

XHELP is the only command to EXEC that doesn't start with CONTROL @EXEC (abbreviation CX).

Why Use It?

XHELP can be extremely useful if you forget a command or correct command syntax. EXEC's help messages are easily available, and they can be more up to date than this manual.

Example

```
) XHELP ↵
... (display list of all EXEC commands) ...
) XHELP ALIGN ↵
... (describes ALIGN command) ...
)
```

End of Chapter

Chapter 4
Choosing a Backup Strategy

This chapter describes the file backup and restore programs and commands available with AOS/VS and AOS/VS II and offers some general guidelines about performing backups and handling storage media.

The major sections of this chapter are

- Comparing Backup Programs
- Backing Up to Magnetic Tape with the DUMP_II and LOAD_II Utilities
- Using a Magneto–Optical Disk as a Backup Medium

Your site should regularly make backup copies of its entire disk file system. Then if files are lost or accidentally deleted, you can restore them from your backup media. How often you back up your system depends on the size of your installation or how often files change. You can do it daily, on alternate days, or weekly. It's prudent to do it daily if possible.

Comparing Backup Programs

Using DUMP/DUMP_II and LOAD/LOAD_II implies a file–oriented approach to backup and recovery. DUMP/DUMP_II can back up specific files and directories, allowing some or all of them to be restored with LOAD or the LOAD_II program. DUMP_II and LOAD_II work with both AOS/VS and AOS/VS II. You can use DUMP_II with labeled or unlabeled tape, with a removable magneto–optical disk treated like an unlabeled tape, or with unlabeled diskettes. You can use DUMP or DUMP_II to copy all user files (a full backup) or only those files that have changed since a given date (incremental backup).

The DUMP_II utility accepts all the same switches that the DUMP command does, and a few more (/ERROR, /MAXCAP, /STAT, and /TAPEMEMORY) that DUMP does not accept. (See the Help files for the DUMP/LOAD commands and the DUMP_II/LOAD_II utilities for a complete description of all the switches; also see the discussion of /TAPEMEMORY in Chapter 5.) Most important, the DUMP_II and LOAD_II utilities offer an option not available with the DUMP command — hard tape error recovery. When a hard error occurs while backing up files, the DUMP_II utility can make it possible to continue the dump operation on the next tape volume, or quitting (stopping) the dump operation.

The DUMP_II and LOAD_II utilities are meant for backup on your local system: the system that produces the material for backup. Some systems provide for remote backup and use a XODIAC network to move copies of files to a larger, central Data General system. The file copies on the central system then serve as the backup. For details on backup over the network, see the latest revision of the manual *Managing and Operating the XODIAC™ Network Management System*.

In addition to or in place of the CLI commands DUMP and LOAD and the utilities DUMP_II and LOAD_II, you may want to consider using the following utilities:

- FSCOPY is an AOS/VS II–only utility that backs up LDUs and recovers LDUs or files. FSCOPY backs up initialized AOS/VS II LDUs, which means that users can continue to work while the backup occurs. FSCOPY provides a consistent backup because it backs up files just as they were when the backup began.

  You can use FSCOPY as your backup/restore utility of choice. You may want to use FSCOPY for full backups and DUMP_II/LOAD_II for incremental backups. FSCOPY is described in Chapter 6.

- LDCOPY is an AOS/VS II–only utility that you can use to back up and recover complete LDUs. It is similar to the PCOPY utility available under AOS/VS. You access LDCOPY from the Disk Jockey utility's Main Menu. There are two versions of LDCOPY: a stand–alone version, which can copy to and from any LDU; and a stand–among version, which can copy to and from any uninitialized LDU (this excludes the master LDU, since, when AOS/VS II is running, the master LDU is initialized). LDCOPY is described in Chapter 7.

- MSCOPY is an AOS/VS–only utility that backs up and recovers modified disk sectors (disk blocks). It can copy only modified sectors: those sectors that have changed since the last full MSCOPY backup. MSCOPY runs while AOS/VS is up; but it can back up only a nonmaster LDU. MSCOPY is most useful for sites that have very large files where relatively few changes occur (like large INFOS II or DG/DBMS database files). It takes a long time to back up these files with DUMP_II or PCOPY; it takes less time to back up only the changed sectors with MSCOPY. MSCOPY is described in Chapter 8.

- PCOPY is an AOS/VS–only utility that you can use to back up and recover complete LDUs. It is similar to the LDCOPY utility available under AOS/VS II. There are two versions of PCOPY: a stand–alone version, which can copy to and from any LDU; and a stand–among version, which can copy to and from any uninitialized LDU (this excludes the master LDU, since, when AOS/VS is running, the master LDU is initialized). PCOPY is described in Chapter 9.

Table 4–1 summarizes the advantages and disadvantages of these approaches.

Table 4–1 Backup/Recovery Approaches: Advantages/Disadvantages

Utility	Advantages	Disadvantages
DUMP_II/LOAD_II	Fastest file–level recovery. Can be used for incremental dumps.	Slower than other options when doing full backups.
FSCOPY	Fastest full backups for AOS/VS II. Applications stopped only briefly.	File–level recovery is not so fast as LOAD_II's.
PCOPY/LDCOPY	Fast backup.	No file–level recovery. LDU must be released.
MSCOPY	Fastest backup for small changes in large AOS/VS environments. Can be used for incremental dumps.	No file–level recovery. LDU must be released. Restricted to certain disks.

Backup Tapes

In order to plan your backup strategy, you need to know how many tapes you will need and how to treat them. The following sections describe tape capacities for a variety of reel–to–reel and cartridge tapes, and also how to care for and use them.

Tape Capacities

The number of reels or cartridges you need for backup depends on the amount of disk space you have, the change rate of your files, the tape length, the recording density, the buffer size used to dump, and the number of tape sets you want to retain. (A tape set includes all volumes of the full backup and all following incrementals, up to the next full backup.) Table 4–2 shows approximate capacities of different tapes.

Table 4–2 Approximate Capacities of Tapes

Description	/BUFFERSIZE[1]	Approximate capacity	
		Mbytes or Gbytes	Disk Blocks
1600–b/in 1200 feet	8192	19 Mbytes	38,912
1600–b/in 2400 feet	8192	38 Mbytes	77,824
6250–b/in 2400 feet	32768	154 Mbytes	315,392
21–Mbyte cartridge	16384	20 Mbytes	40,960
130–Mbyte cartridge	16384	74 Mbytes (start–stop) or 130 Mbytes (streaming)	151,552 266,240
150–Mbyte QIC	16384	139 Mbytes	284,672
320–Mbyte QIC	16384	296 Mbytes	606,208
525–Mbyte QIC	16384	486 Mbytes	995,328
5–Gbyte 4mm DAT	32768	5 Gbytes[2]	10,485,760
2–Gbyte 8mm cartridge	32768	2 Gbytes	4,194,304
5–Gbyte 8mm cartridge	32768	10 Gbytes with DENSITY=HIGH[2]	20,971,520

[1] Data General DS/7500 and ECLIPSE MV/1000 DC, MV/1400 DC, MV/2000 DC, and MV/2500 DC systems allow a maximum buffer size of 16384.

[2] These drives use data compression, so actual capacity can vary depending on data.

To get the approximate number of tape volumes needed for a full backup of any LDU, type SPACE and the specific ldu–name and divide the CUR figure by the pertinent capacity in blocks figure. For the master LDU, subtract 20,000 blocks — for system files — from the CUR figure before dividing. Please remember that these figures are approximate. DUMP_II requires some overhead for storing directory and file information.

For incremental backup, take 20% of the number of full backup tapes. Multiply this number by the number of incremental backups you plan between full backups. Add the total incremental backup number to the full backup number. This total gives you the approximate number of tapes you need for one backup set.

You should have two or more backup sets of tapes. With two sets, you can keep the last backup set intact and use the previous backup set for the new backup. Ideally, you'd create an incremental backup tape set every day, and a full backup tape set (including system disk) once a week.

Remember that dump density and label density must match; if they do not, you will get the message that there is a density mismatch. So plan ahead. If you want to dump at 6250 b/in, prepare labeled tapes by setting the hardware panel switch to 6250 or use the LABEL program's /DENSITY switch (X LABEL/DENSITY=6250).

Tapes and diskettes should have paper labels on them on which you should write the date, volume ID, and other information you might need to restore the material.

Before Starting a Backup

Before starting a backup, be sure you have enough time and ample backup media. Whether you use the CLI DUMP command or the DUMP_II program, make sure that all important files are closed during a dump. (If you run the AOS/VS II FSCOPY program, system activity need be paused for only a short while.) If a file is open for write access when dumped, changes may have been made that have not yet been written to disk. This means that the file dumped is not current and can make the backup useless. Ideally, during backups, all timesharing users will be logged off. All CEO, DG/DBMS, or INFOS II processes should have been shut down normally. If abnormal shutdown has occurred since the last backup, be sure that recommended verification programs (such as IVERIFY for INFOS II) have been run to ensure the integrity of databases.

Storing and Handling Tapes

If you handle and store your reel–to–reel and cartridge tapes properly, they will last a long time and the data stored on them will remain intact indefinitely. Some handling cautions and hints follow.

- Cold, heat, and dryness can harm tapes. Store tapes between 50 and 90 degrees Fahrenheit (10 and 40 degrees Centigrade). Use tapes between 60 and 90 degrees Fahrenheit. Relative humidity should be 20% – 90%. If you move a tape to a warmer or colder place, give it some time (ideally, 24 hours) to adapt to the temperature change before using it.
- Store tape reels in their outer plastic covers; remove the cover from the tape just before you use it. Store cartridge tapes in their cases.
- A magnetic field can erase part or all data on a tape. Keep tapes away from electric motors, magnets, and transformers.
- Always handle the tape reel by its hub (center). If you hold the outer flanges, you might squeeze them — compressing and damaging tape edges. Edge damage is a common cause of tape failure.
- If a tape has no paper label, apply one. Write on the label before applying it to the reel, to avoid bending reel flanges. If a label is already on a reel, use only a felt–tipped pen. (Never use pencil: graphite is a conductor and can cause problems.)
- Don't allow tape to rest or drag on the floor. Dust on the tape could prevent data transfer from the heads.
- Never touch the tape in the data area. Touch only the tape that precedes the beginning–of–tape (BOT) marker or follows the end–of–tape (EOT) marker. These reflective markers define the bounds of data. They are adhesive foil strips on the back (nonrecording surface) about 16 feet after the beginning and 25 feet before the end of the tape. If you touch the data area, the oil on your finger could make that part of the tape unreadable.
- Don't bend or twist tape when threading it (if your units have reels). Bending can dislodge the magnetic coating, causing data loss.
- Before covering a tape reel, place a foam retainer or plastic strip on the leading edge of the tape. This retainer will prevent the tape from unwinding in storage.
- Tapes last a long time, but eventually they wear out. A growing number of *Soft Error* messages are a warning. (A few *Soft Error* messages are normal.)

Using a Magneto–Optical Disk as a Backup Medium

Except where using diskettes is required, you will probably use reel–to–reel or cartridge tape as the backup medium of choice. Tapes are relatively inexpensive, long–lasting, and hold a great deal of information. An effective alternative to tape is the a magneto–optical disk like the Model 6627 disk.

The Model 6627 disk is a 5–1/4 inch rewritable and removable magneto–optical disk that holds 590 Mbytes, 295 Mbytes on each side. While slower than other disks, the magneto–optical disk is approximately the same speed as tapes, and its portability and high reliability make the magneto–optical disk an excellent backup medium.

You can use this disk as a backup medium in two ways, as an LDU or as an unformatted medium. (Backing up to the disk as an unformatted medium is faster.)

Used as an LDU, initialize the formatted disk using the CLI INITIALIZE command (for example, INITIALIZE @DPJ11). Then, you can use disk mirroring, FSCOPY, LDCOPY, PCOPY, or the CLI MOVE command to produce a backup. Using any of the disk–based approaches requires careful planning because the disks or LDUs must be the same size for mirroring or copying to be possible. If you try to copy files using MOVE, and require more space on the receiving disk than actually exists, you will get the message *Control point directory max size exceeded*, and you will have to start over. But while writing the backup may be slower, accessing the files is faster than reloading from tape because of the random access nature of the disk.

Used as an unformatted medium, insert a disk and write to it using the format

DUMP_II*[/switches]* @DPJn *[pathname] [...]*

DUMP_II writes to the disk sequentially, as though it were an unlabeled tape, prompting you to mount the next volume after it fills each side of a disk. When prompted, remove the disk, turn it over, and reinsert it. Otherwise, proceed exactly as you would when writing multi–volume unlabeled tapes. You must, therefore, be careful to affix labels to each disk. When you reload from the backup set using LOAD_II, you must be careful to mount the disk volumes (sides) in the right order. Also, you should use the /BLOCKCOUNT=64 switch to maximize performance.

End of Chapter

Chapter 5
Using DUMP/DUMP_II and LOAD/LOAD_II to Back Up and Restore Files

This chapter describes the file backup and restore commands DUMP and LOAD and the utilities DUMP_II and LOAD_II. The utilities run under both AOS/VS and AOS/VS II. Only the 16–bit CLI DUMP and LOAD commands support labeled diskettes. For a general introduction to backup, see Chapter 4.

The major sections of this chapter are

- Backing Up to Magnetic Tape with the DUMP_II and LOAD_II Utilities
- Using DUMP_II and LOAD_II with High–Capacity Cartridge Tapes
- Backing Up a Mirrored LDU
- Backup Macros for Tape
- Restoration Macro for Tape
- Hard Tape Error Recovery with the DUMP_II and LOAD_II Utilities
- Backing Up to Diskettes with the DUMP and LOAD Commands
- Backup Macros for Diskettes
- Restoration Macro for Diskettes

Using DUMP/DUMP_II and LOAD/LOAD_II implies a file–oriented approach to backup and recovery. DUMP/DUMP_II can back up specific files and directories, allowing some or all of them to be restored with LOAD or the LOAD_II program. DUMP_II and LOAD_II work with both AOS/VS and AOS/VS II. You can use DUMP_II with labeled or unlabeled tape, with a removable magneto–optical disk treated like an unlabeled tape, or with unlabeled diskettes. You can use DUMP or DUMP_II to copy all user files (a full backup) or only those files that have changed since a given date (incremental backup).

The DUMP_II utility accepts all the same switches that the DUMP command does, and a few more (/ERROR, /MAXCAP, and /STAT) that DUMP does not accept. (Read the manual *Using The CLI (AOS/VS and AOS/VS II)* for a complete description of all the switches for the DUMP/LOAD commands and the DUMP_II/LOAD_II utilities.) Most important, the DUMP_II and LOAD_II utilities offer an option not available with the DUMP command — hard tape error recovery. When a hard error occurs while backing up files, the DUMP_II utility makes it possible to continue the dump operation on the next tape volume, or quitting (stopping) the dump operation.

The DUMP_II and LOAD_II utilities are meant for backup on your local system: the system that produces the material for backup. Some systems provide for remote backup and use a XODIAC network to move copies of files to a larger, central Data General system. The file copies on the central system then serve as the backup. For details on backup over the network, see the latest revision of the manual *Managing and Operating the XODIAC™ Network Management System*.

Backing Up to Magnetic Tape with the DUMP_II and LOAD_II Utilities

This section gives some pointers on using the DUMP_II utility with tape, and describes some macros to help you. All comments on the DUMP_II utility apply to the DUMP command unless otherwise noted. To use the DUMP command with diskettes, read the section on "Backing Up to Diskettes with the DUMP and LOAD Commands."

The DUMP_II and LOAD_II utilities are very versatile. They use switches that are fully detailed in the manual *Using the CLI (AOS/VS and AOS/VS II)*. The DUMP_II utility accepts all the switches that the DUMP command does; it produces the same output as the DUMP command. The LOAD_II utility can load tape files dumped with either DUMP or DUMP_II. Finally, the DUMP_II/LOAD_II utilities offer hard tape error recovery, described in the section "Hard Tape Error Recovery with the DUMP_II and LOAD_II Utilities" later in this chapter.

The DUMP_II and LOAD_II utilities accept templates. The system matches templates, dumping or loading the files in the order it finds them, not necessarily in the order you expect. (The CLI FILESTATUS command behaves in the same way.)

The DUMP_II utility, without a directory name template, copies all files and directories in the working directory to a dump file. It writes a header before each file, with the file's name, date of creation, and other data, followed by a trailer after each file that records the ACL and any UDA information. Unless you use the /FLAT switch, DUMP_II maintains the directory structure. Later if you load the dump file into the directory from which DUMP_II was executed, the system will try to recreate the original structure. It's very important that the working directory be the same for the load operation as it was for the dump operation. The root directory (:) is the best directory from which to start system–wide backups and restorations.

The dump file (which becomes the input file for load) can be a labeled tape file like ROOT or LDU2, or a tape file number like @MTB0:3. For tape, we strongly recommend labeled tapes, although they are not mandatory, because labeled tapes allow a dump to span more than one tape volume with better data integrity. In a production environment, it is critically important to use labeled tapes.

You can label tapes from any CLI process using the LABEL program at any time. Use the tape unit name; then the label (maximum length six characters). For example,

```
) X LABEL @MTB0 FULL00 ↵
```

Both the DUMP/LOAD commands and the DUMP_II/LOAD_II utilities work well with disk mirroring, an approach that maintains logically and/or physically identical copies of an LDU. See the section "Backing Up a Mirrored LDU," later in this chapter.

Handling Hard Errors

A *Hard Error* or *Physical Unit Failure* message means that the rest of the tape cannot be read from or written to if you are using the CLI DUMP or LOAD commands. If this message appears during backup you must use another tape. If it appears during a restoration, restart the restoration. If the hard error message recurs, this probably means you cannot restore data from the rest of the this tape and from subsequent tapes in this set. Try the tape on another unit if you have more than one tape unit, or clean the tape unit heads with an alcohol–soaked swab or cleaning cartridge.

However, the DUMP_II/LOAD_II utilities provide you with the capability to recover from hard tape errors. When a hard error occurs while backing up files, DUMP_II lets you quit or mount the next volume. When a hard error occurs on a load, LOAD_II lets you either retry reading from the bad block that caused the error, advancing to the next good block and reading, or quitting the load operation.

CAUTION: *In rare cases, most cartridge tape drives may generate the message* Fatal buffered tape error.

DUMP_II and LOAD_II cannot recover from this kind of error. If you get this error, discard the tape and do not reuse it. Restart the dump. See the Notes and Warnings section of the release or update notice for the latest status of this problem.

Using DUMP_II and LOAD_II with High–Capacity Cartridge Tapes

There are some special considerations when using DUMP_II/LOAD_II with high–capacity SCSI–2 cartridge tape drives. The following sections will help you use your drives to the best advantage.

Using the TAPEMEMORY Feature to Stream SCSI–2 Helical Scan Tape Drives

Whenever possible, it is better to run a tape unit in streaming mode than start/stop mode. Streaming a tape drive is likely to make the drive last longer. You can stream some cartridge tape drives using the DUMP_II /MAXCAPACITY switch. You cannot use the /MAXCAPACITY switch, however, to stream SCSI–2 helical scan tape drives. To stream these tape drives, use the /TAPEMEMORY switch instead. The switch has the following format:

/TAPEMEMORY=MAX or /TAPEMEMORY=n

Use this switch to stream SCSI–2 helical scan cartridge tape drives, like the 4–mm DAT (Model 6762) and 8–mm Models 6760/6761, in order to prolong the life of these drives. While this switch helps stream the tape drive, it is not intended to improve performance nor does it provide any benefit with other drives.

Use this switch for large backup operations only when the system is not active: the switch sets aside n (1–250) Mbytes of memory (or a maximum that it computes) for its buffers, and runs DUMP_II as a resident process. You must have the privilege Change [process] type and sufficient physical memory to use this switch.

If you use /TAPEMEMORY=MAX, DUMP_II chooses a maximum value for you. If you use /TAPEMEMORY=n, the rule is that each Gbyte of data to be dumped requires 15 Mbytes of memory. In addition to the /TAPEMEMORY switch, you should also use /BLOCKCOUNT=255 and /BUFFERSIZE=n, where n is the maximum buffer size for the tape drive that is supported by your system. DUMP_II will disregard the value you choose if there is not enough memory available or if that value would prevent the system from running. It will inform you of this, and continue to run with a smaller value.

NOTE: If you use /TAPEMEMORY=MAX, run only one dump process or you may hang the dump processes or the system. You can run multiple processes if you use /TAPEMEMORY=n, but whether you run one or several processes ensure that the summation of n for all processes is the smaller of two values: 80% of physical memory or physical memory minus 5 Megabytes.

Selecting Tape Density with 8–mm Tape Drives

By default, 8–mm tape drives set tape density to HIGH with new tapes or to the density last used. If you want to ensure that you always write to the tape with high density set, you can either use VSGEN to set the default density to HIGH, or you can specify /DENSITY=HIGH on the command line or in macros that perform backups.

Tape Interchange Between Model 6590 and Model 6760/6761 Tape Drives

SCSI–2 Model 6760/6761 cartridge tape drives compress data by default. SCSI Model 6590 cartridge tape drives do not compress data. You can interchange data between these two drives so long as the data is not compressed. To write to a tape on Model 6760/6761 cartridge tape drives with data compression turned off, use the DUMP_II /NCOMPRESS switch. Also use the switch /DENSITY=LOW. If you wish to use a SCSI–2 Model 6760/6761 cartridge tape drive to append to a tape containing files written by a Model 6590, you must use the /NCOMPRESS switch.

NOTE: Only DUMP_II/LOAD_II and DUMP_3/LOAD_3 support turning off data compression. Other utilities (for example, CPIO_VS, DDUMP, and TAR_VS) do not let you turn off data compression.

Tape Interchange and Buffer Size

Most systems let you create tapes using a buffer size of 32768, while some other systems have a maximum buffer size of 16384. If you try to load a tape written with a buffer size of 32768 on systems with the smaller buffer size, the load will fail. The systems that have a maximum buffer size of 16384 are Data General DS/7500 and ECLIPSE MV/1000™ DC, MV/1400™ DC, MV/2000™ DC, and MV/2500™ DC systems.

Tape Labeling

Magnetic tapes are either labeled or unlabeled. An unlabeled tape contains no label information. A labeled tape has information including a volume ID (volid) for the tape and a filename and expiration date for the tape fileset. DUMP_II/LOAD_II, for example, are distributed on unlabeled tape with an operating system update or release.

Components of Labeled Tape

There are several different kinds of labels written to a labeled tape. The most important ones are

- The volume header label (contains the volid and is created by the LABEL utility); and
- The first file header label (HDR1, created by the system when it writes a file to the tape).

A labeled tape, after it has been labeled via LABEL and DUMP_II has written user files to it, has the structure shown in Figure 5–1. Compare with the section "Components of Labeled Tape" in the chapter on labeled tape in *Using the CLI*, which describes all possible fields.

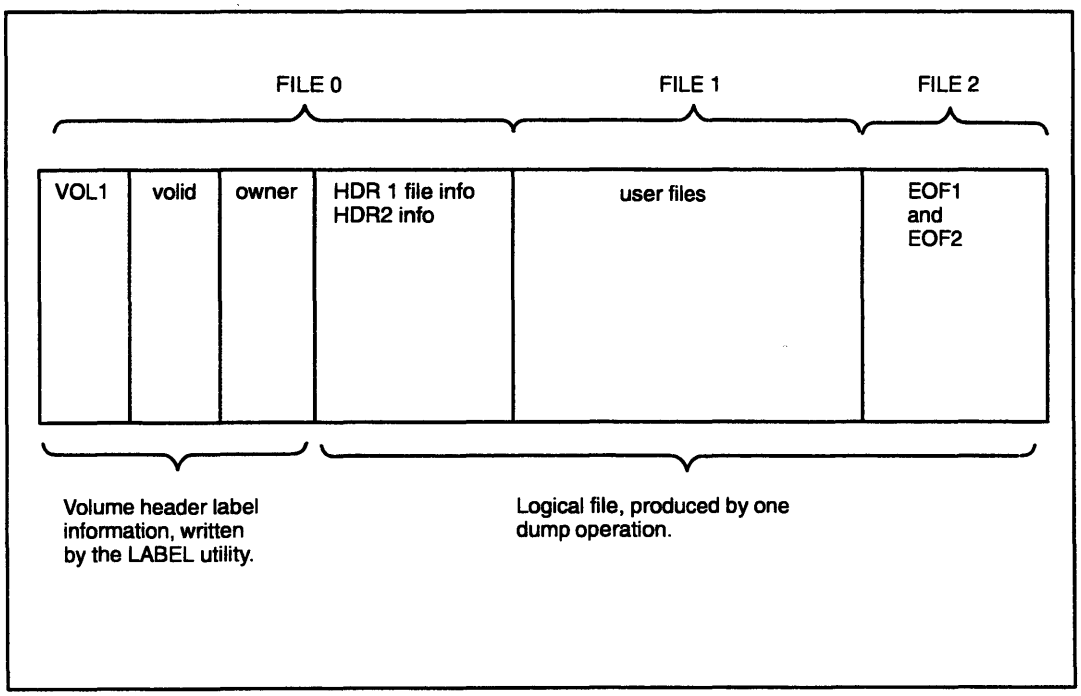


Figure 5–1 Information on a Labeled Tape

The fields shown on the tape have the following meanings.

Field	Description
VOL1	is a fixed string that means volume type 1. The LABEL utility always writes VOL1 on every tape.
volid	is the volume ID assigned by a person using the LABEL utility (X LABEL unitname volid). The maximum length is six characters.

Field	Description
owner	is an optional owner field, whose contents users may specify with the LABEL /OWNER= switch. The maximum length is 14 characters for ANSI label tapes and 10 for IBM label tapes. If you use the /OWNER switch with LABEL, you must also use the /OWNER switch with DUMP_II. The DUMP command does not have an /OWNER switch but retains the string from the LABEL utility.
HDR1 file info	is written by the system during the write operation. It contains the following items. All items are created and used by the operating system only, unless noted otherwise. • filename for the fileset, as supplied by the person who started the tape operation. (This name has no relationship to the files actually written to the tape.) For example, the command DUMP_II/V TAPE:USERS creates the filename USERS on the tape. The maximum filename length is 17 characters; • fileset ID; • file section number; • file sequence number; • generation number and version number (created by the operating system but not used by DUMP/DUMP_II or LOAD/LOAD_II); • creation date; • expiration date (the default is 90 days from the creation date; the DUMP/DUMP_II /RETAIN switch can override the default); • block count (always 0 in HDR1); • operating system ID (as set by SYSID command, or the default).
HDR2 info	is written by the system. HDR2 contains the record format specifier (a one–letter code), block length (buffer size), and a code for record length.
user files	is a dump file containing all of the disk files the DUMP_II utility or the DUMP command writes to the tape.
EOF1 and EOF2	At the end of each logical file written, the system writes a label with EOF1 and EOF2. It writes an EOV1 and EOV2 label at the end of each reel. (There is no EOV label if the file fits on a single reel.) EOF1 and EOV1 each record the number of blocks written; this number serves as an error check when the tape is read.

DG, ANSI, or IBM Format?

By default, labeled tapes are written in DG format. (Labels created by both the CLI DUMP command and the DUMP_II utility are compatible with the American National Standard for Magnetic Tape Labels (ANSI x3.27–1978) Level 3.) To write labeled tapes that will be read on a DG, ANSI, or IBM (EBCDIC) system, proceed as follows.

- If the destination system is a DG system (AOS/VS or AOS/VS II), use the LABEL program without the /I switch. To write to the tape, you can use any CLI command or the default format in any write statement.
- If the destination system is an ANSI–based, non–DG system, use the LABEL program without the /I switch. To write to the tape, you can't use CLI commands. You must use a program that either opens the labeled tape (?OPEN call to @LMT:volid:filename) for ANSI format; or use a program that employs the high–level language equivalent of ?OPEN/?WRITE to specify ANSI format.
- If the destination system is an IBM system (EBCDIC), use the LABEL program with the /I switch. You can then write to the tape using the DUMP_II utility or the CLI command DUMP with the /IBM switch. On an IBM system, you can then load the tape and use it with IBM syntax. (You can load labeled tapes produced on an IBM system in IBM format by adding the /IBM switch to the LOAD_II program or the CLI LOAD command.

Unlabeled Versus Labeled Tape

The whole approach to labeled tape differs from the approach to unlabeled tapes. For example, typical I/O to unlabeled and labeled tape might look like this.

Unlabeled Tape	Labeled Tape
DIR :	MOUNT/VOLID=xxx TAPE PLEASE
DUMP_II @MTB0:0 +	DIR :
DUMP_II @MTB0:1 UDD:SALLY:#	DUMP_II :UDD:OP:TAPE:USERS UDD:#
DUMP_II @MTB0:2 UDD:JACK:#	
. . .	. . .

NOTE: Load sequence would be the same, just substitute the LOAD_II utility for the DUMP_II utility.

The labeled approach won't work on unlabeled tapes, but the unlabeled approach can work on labeled tapes, because the tape labels are individual tape files. Although labeling tapes requires extra effort, labeled tapes offer many advantages.

- The tape itself contains information about material stored on it: user label text, filename, and expiration date.
- Users can create tapes to be read on other operating systems, like an IBM system or a system that uses ANSI–standard labeled tapes.
- Users have control over the amount of time each tape file will be retained, via the DUMP/DUMP_II /RETAIN= switch. The system will not overwrite files dumped to a labeled tape until the retention period has passed or until the tape is relabeled.
- Data General data management programs — INFOS II and DG/DBMS — have logging utilities that expect labeled tapes.

Planning and Assigning Tape Labels

Generally, tape labels must be consistent to be worthwhile. There's a big difference between typing

```
) DUMP_II/V @MTB0:3 +.F77 ↵
```

and

```
) DUMP_II/V MYTAPE:F77_SOURCES +.F77↵
```

The first approach is hardware oriented: people must know the unit number and tape file number they want. The second approach is software oriented: people can use any link name and a real filename up to 17 characters instead of the unit and file number. If people try to combine the labeled and unlabeled approaches, they may get confused.

Tape labels must be planned to be effective. You may not want to let users create their own labels because you will have to keep track of their labels and tapes. If your installation lets users do their own labeling, there should be a standard for the labels, and you, as the system operator or manager, will have to administer a storage system for the user–labeled tapes so you can find them easily. If users can't create their own labels, you may want to prepare many empty, labeled tapes for them with the LABEL utility.

The industry standard length of labeled tape volume IDs (volids) is six uppercase, alphanumeric characters. So the volids you choose should be six characters or less, yet be as descriptive as possible.

NOTE: For multi–volume labeled tape sets, you must make the volid of each tape volume unique.

You might choose an arrangement in which the leading characters specify the usage type for the tape, and the trailing characters are numbers that give the tape sequence. For example,

volids GP0000 through GP9999 are for General Purpose (GP) user use;

volids DB0000 through DB9999 are for DataBase (DB) archiving.

Or, you could use the date and sequential numbers; e.g., 130100, 130101, etc. for 13 January. If you alternate daily dumps, use a suffix of A or B. If you have multiple systems, you may want to include a short system identifier.

After deciding on the volids, you can use the LABEL utility to prepare prelabeled tapes for each use type. For example,

Put a new tape on unit MTB0. When it is on line and ready, type

```
) X LABEL @MTB0 GP0000↵
```

Dismount the tape and put another new tape on the unit, and type

```
) X LABEL @MTB0 GP0001 ↵
```

A tape labeled with the LABEL program has the label at the beginning, but users don't need to know about it since they write to the tape and read from it by link name and filename. The link name is the name of the link file created when the user issues the MOUNT command.

The tape label includes the volume ID, owner (if specified), system name, and revision. If the tape has been written to, the label also contains the expiration date and fileset ID. Labels written by the LABEL utility are in ASCII by default, so you can use the CLI command TYPE to read them; e.g., TYPE @MTB0. (All LABEL utility syntax and switches are further described in the manual *Using the CLI (AOS/VS and AOS/VS II).*)

If you're planning to dump at a nondefault density, be sure to label the tape using that specific density. For example, use the /DENSITY= switch to the LABEL utility:

```
) X LABEL/DENSITY=6250 @MTD0 VOL1 ↵
) MOUNT/VOL=VOL1 MYTAPE ↵
) DUMP_II/V/L=LFILE/DENSITY=6250 MYTAPE ↵
```

NOTE: If you run a locked CLI on the system console, it will not obey the command X LABEL. You must either unlock the system console or use another terminal to label tapes.

After using the LABEL command to label a tape, you should write the label name on an adhesive label and stick it on the tape reel or cartridge. Later when someone needs a blank tape, you can get the next sequential reel and mount it on a unit; then use the SEND command to tell the person the volid. After the person finishes with the tape, you can file the tape with the person's name so that you can find it easily.

When dumping files to labeled tape it is a good idea to use the /RETAIN switch as an argument to DUMP/DUMP_II, like this:

```
) DUMP_II/V/RETAIN=14 TAPE:SOURCES +.PL1 ↵
```

Then the system won't overwrite file SOURCES until 14 days have passed. This gives users great control over backup periods and can be very helpful for file security. Anyone, however, can overwrite the label (with new retention period and other information) by relabeling the tape with the LABEL utility or with a CLI command or other utility that writes to unlabeled tapes.

Backing Up a Mirrored LDU

When you use DUMP_II to back up a mirrored LDU, you should release the LDU to make sure that files are closed. However, you can immediately reinitialize one image using the command INITIALIZE/NOMIRROR, and continue to run, thus improving system availability. This procedure can take less than a minute.

If you use DUMP_II, you must initialize the image in a different directory before you can back it up. (You need to initialize the image in a different directory because you can't initialize two LDUs with the same name in the same directory.)

After you have grafted the LDU onto an appropriate directory, you can then use DUMP_II to perform the backup.

When the backup is finished, release the LDU and make the root directory your working directory. Initialize the two images (if both were released and one was not reinitialized), or initialize (with /NOMIRROR) one image and then the other (with MIRROR/SYNC).

On AOS/VS II systems, you may want to consider using FSCOPY instead, especially if you are using mirroring for backup and not for availability.

Example of Backing Up a Mirrored LDU

Backing up a mirrored LDU requires a brief pause in system activity. Because you want to make sure that files are closed, shut the system down, even though briefly. Users must exit from their text editors. If you are running a CEO system, shut it down also. As soon as all files are closed and all users are disabled (use the CX DISABLE/ALL command) from logging on to the system, release the LDU.

When you use DUMP_II for backup, you can initialize (using the /NOMIRROR switch) one of the images, restart the CEO system and notify users that they may safely resume their work. Then after this brief pause, you can start the backup.

Here is an example of a procedure and dialog you could use to back up a mirrored LDU image using DUMP_II.

1. Inform people that the system will be down briefly at a specific time.

2. Just before the appointed time, prevent users from logging on by issuing the command

) CX DISABLE/ALL ↵

3. Ask users to close their files, or bring down the CEO system.

) BROADCAST About to do a backup. Please log off. ↵

   or

) BROADCAST CEO coming down. Please log off.↵

4. After a few minutes, if you run the CEO system, shut it down.

) CEO.SYSTEM STOP ↵

5. When the WHOS.CLI macro (shipped with the operating system) shows that the system is inactive, proceed.

6. Turn on the Superuser privilege on and make the root your working directory.

) SUPERUSER ON ↵

   Su) DIRECTORY : ↵

7. Release the LDU.

   Su) RELEASE UDD1 ↵

8. Initialize one of the UDD1 images:

   Su) INITIALIZE/NOMIRROR @DPJ1 ↵

9. Enable all terminals.

   Su) CX ENABLE/ALL ↵ — Users will see the log–on banner on the screen and can resume their work.

10. Make a directory other than the root your working directory. In our example, the directory :BACKUP is that directory. Type

    Su) DIRECTORY :BACKUP ↵ — Move to a different directory.

11. Initialize the second UDD1 image.

    Su) INITIALIZE/NOMIRROR @DPJ2 ↵ — Initialize DPJ2 with /NOMIRROR.

12. Perform the backup using DUMP_II.

    ... (time passes) ...

13. After the backup is finished restart DPJ2 as a synchronized mirror with these commands:

    Su) RELEASE UDD1 ↵ — Release DPJ2.

    Su) DIRECTORY : ↵ — Make the root your working directory again.

    Su) MIRROR/SYNC UDD1 @DPJ2 ↵ — Start the resynchronization process.

    ... (time passes) ...

    Done!

LDU image DPJ2 of the LDU named UDD1 is now synchronized.

Backup Macros for Tape

Macros for full and incremental dumps to tape are shown in Figures 5–2 and 5–3, followed by an example of each. A macro to restore files from tape is shown in Figure 5–4, followed by an example of a restoration from tape. Data General shipped these macros with the operating system to ease backup. You must change a specified character as described in each macro (type FULL_DUMP) before it will work.

The macro FULL_DUMP.CLI, Figure 5–2, does a full backup of all initialized LDUs, excluding system–only directories. INC_DUMP.CLI, Figure 5–3, does an incremental backup of all initialized LDUs. INC_DUMP.CLI dumps only those files created or modified since the last backup based on the file LAST_DUMP_DATE, which contains the date and time of the last backup. The backup macros establish tape volume IDs for each dump and prompt the system operator through each step. They specify a volume ID list of 10 tapes and allow this volume ID list to be extended.

A dump can back up fewer than 8 Mbytes of material (on one 400–foot tape) or include as many as 800 Mbytes at 1600 b/in or 3.2 Gbytes at 6250 b/in (on 21 2400–foot tapes). The limiting factor is the volume ID list, which can't exceed 128 characters for a single MOUNT command. Volume IDs of less than six characters would allow more volumes to be specified.

You can use these macros — or expanded versions of them — at your own site. Each macro dumps with the /RETAIN=0 switch. This switch allows the tape set to be reused *immediately*, if desired, without relabeling. You should use a nonzero retention period, for example, /RETAIN=28 for full backups and /RETAIN=7 for incremental backups.

If you have MTB or MTD tape units, you might want to specify the highest density in the DUMP_II commands. To do this, use the /DENSITY=n switch. With an MTD unit, specify a buffer size of 32768 (/BUFFERSIZE=32768) in the DUMP or DUMP_II command.

The macro FULL_DUMP.CLI in Figure 5–2 below, does a full backup of all initialized LDUs, excluding system–only directories.

The FULL_DUMP.CLI Macro

```
[!EQUAL,1,2]
[!equal,comment,]
This macro does a full backup of the entire system, excluding system
directories and files, to labeled tape, in batch. It also sets up the
labeled tape volume IDs to be used. The tape volume IDs are FULLnn,
where nn is the sequence number of the tape in the backup.

The tape fileset name used for the backup is ROOT. The macro also
explains how to specify a backup to a second fileset (like UDD), if your
system is large enough to make a second tape fileset worthwhile.

This template used for the backup excludes DG–supplied directories. You
can restore these from your system SYSTAPE and from the INC_DUMP tapes.

A file named LAST_DUMP_DATE is needed in the root directory to start
things off. If it doesn't exist when you run this macro, the macro will
create it.

To execute the macro, the person acting as system operator must go to a
user terminal, log on as a Superuser (like OP), and type the macro name.
The macro will then help him/her through the procedure. Tape mounts
and CONTROL @EXEC MOUNTED commands at the system console will be needed.

The command and pseudomacro syntax used is explained in the manual
Using the CLI.
[!end]

[!equal,comment,] Make sure EXEC operator mode is on. [!end]

[!nequal,(ON),([!operator])]
  write Error – Operator is not on. Please go to the system console and
  write type [!read CONTROL @EXEC OPERATOR ON NEW LINE. Return here and
  write press NEW LINE.]
[!end]

[!equal,comment,] Check for file LAST_DUMP_DATE. If it doesn't exist,
tell user, then create it. [!end]

push
dir :
superuser on
```

Figure 5–2 FULL_DUMP.CLI Macro for a Labeled Tape Dump (continued)

```
[!nequal,([!filenames LAST_DUMP_DATE]),()]
   [!equal,([!filenames DATE_DUMP_STARTED]),()]
      write  Last backup was done on ,,, [LAST_DUMP_DATE]
             write/l=DATE_DUMP_STARTED  [!date]:[!time]
   [!else]
      write Full backup was started at ,,, [DATE_DUMP_STARTED]
   [!end]
[!else]
   write File LAST_DUMP_DATE doesn't exist! I am creating a
   write LAST_DUMP_DATE file that specifies today's date and the
   write current time as follows: [!date]:[!time]
   write If you don't do a full backup today please delete LAST_DUMP_DATE
   write/l=LAST_DUMP_DATE  [!date]:[!time]
[!end]

write
write  *** ,, Full backup of directory [!dir] on [!date]:[!time] ,, ***
write
[!equal,comment,] Get person's name and write to file DUMPERS_NAME.[!end]

delete/2=ignore  :DUMPERS_NAME
write/l=:DUMPERS_NAME  [!read  Please type your name: ]
pop
dir/i

[!equal comment,] Post the needed mount and dump commands in batch.
[!end]
qbatch/m/operator
superuser  on
dir  :

[!equal,comment,]  Issue a mount command for 10 labeled tape volumes.
The volids for full backup are  full00, full01,...full09. (First, to
prevent problems, delete any linkname of the name MYTAPE in user's
directory.)[!end]

delete/2=ignore   :udd:[!username]:MYTAPE

mount/extend/volid=full00/volid=full01/volid=full02/volid=full03&
/volid=full04/volid=full05/volid=full06/volid=full07/volid=full08&
/volid=full09  MYTAPE  Please mount the longest tape you can.
```

Figure 5–2 FULL_DUMP.CLI Macro for a Labeled Tape Dump (continued)

```
[!equal,comment,] Now back up to the labeled tape file. The fileset name
used here is ROOT. The template does not exclude UDD or any nonmaster
LDUs. If you want a backup separate from ROOT (like UDD), exclude the
directory(s) via the backup template. Then insert a "mount/volid=dirnn"
and "dump ...  dir:#" command after the  dismount command. For example,
to exclude UDD from the ROOT backup and include it in a separate backup:
1. Add the text  \UDD:#  to the ROOT backup template.
2. Add commands  mount/volid=UDD00/vol=UDD01... MYTAPE Please  and
            backup ... MYTAPE:UDD  UDD:#\?+.BRK\+.ED\+.LS\?+.TMP
  after the dismount command.

The listing file (/l) in batch is the line printer. [!end]

write/l  This is a full backup of directory [!dir] started at
write/l  [!date]:[!time] by [DUMPERS_NAME]

dump_II/v/l/buffersize=8192/retain=0   &
   :UDD:[!username]:MYTAPE:ROOT              &
   #\HELP\NET\PAGE\PATCH\PER\PROC\QUEUE\SWAP\SWAP.SWAP&
   \SYSGEN\UTIL\?+.BRK\+.ED\?+JOB\+.LS\?+.TMP  NET:NETGEN:+

dismount  MYTAPE  Full backup of directory [!dir] is done.
pause 5
send 2 ,, Check the printer — file [!username].OUTPUT.n  — for errors.
send 2 ,, A list of files backed up is printed in file [!username].LIST.n

[!equal,comment,] Delete and rename file LAST_DUMP_DATE... but do it
only if file DATE_DUMP_STARTED exists. This prevents LAST_DUMP_DATE
from being deleted if the batch job was aborted. The terminating )
is needed for qbatch/m syntax. [!end]

[!nequal,([!filenames DATE_DUMP_STARTED]),()]
 delete/2=ignore  LAST_DUMP_DATE
 rename/2=ignore  DATE_DUMP_STARTED  LAST_DUMP_DATE
[!end]
)
[!ELSE]
 write This macro is nonexecutable. To make it executable you must use
 write a text editor to change the 2 in line 1 to 1 — both numbers
 write must be 1.
 write
 write Backup is very important. Before using the edited macro
 write for backup you should understand how it works. Please test this
 write macro. Use it to back up files. Then try restoring files using
 write the RESTORE_TAPE macro — before relying on this macro
 write routinely.
[!END]
```

Figure 5–2 FULL_DUMP.CLI Macro for a Labeled Tape Dump (concluded)

The INC_DUMP.CLI Macro

```
[!EQUAL,1,2]
[!equal,comment,]
This macro does an incremental backup of the entire system, excluding
system directories and files, to labeled tape, in batch. It also sets up
the labeled tape volume IDs to be used. The tape volume IDs are INCRnn,
where nn is the sequence number of the tape in the backup.

The tape fileset name used for the backup is ROOT. The macro also
explains how to specify a backup to a second fileset (like UDD), if your
system is large enough to make a second tape fileset worthwhile.

This template used for the backup includes DG–supplied directories. You
can restore these from your system SYSTAPE and from backups produced by
this macro.

A file named LAST_DUMP_DATE is needed in the root directory to start
things off. If it doesn't exist when you run this macro, the macro will
recommend recovery steps and stop.

To execute the macro, the person acting as system operator must go to a
user terminal, log on as a Superuser (like OP), and type the macro name.
The macro will then lead him/her through the procedure. Tape mounts
and CONTROL @EXEC MOUNTED commands at the system console will be needed.

The command and pseudomacro syntax used is explained in the manual
Using the CLI.[!end]

[!equal,comment,] Make sure EXEC operator mode is on. [!end]

[!nequal,(ON),([!operator])]
  write Error – Operator is not on. Please go to the system console and
  write type [!read CONTROL @EXEC OPERATOR ON newline. Return here and
  write press NEW LINE]
[!end]

[!equal,comment,] Check for file LAST_DUMP_DATE. If it doesn't exist,
skip to end, give recovery advice, and stop. [!end]

push
dir :
superuser on
```

Figure 5–3 INC_DUMP.CLI Macro for an Incremental Labeled Tape Dump (continued)

```
[!nequal,([!filenames LAST_DUMP_DATE]),()]
 [!equal,([!filenames DATE_DUMP_STARTED]),()]
     write  This backup will back up all copyable files created or
     write  modified since [LAST_DUMP_DATE]
            write/l=DATE_DUMP_STARTED   [!date]:[!time]
 [!else]
     write The last incremental backup was started at [DATE_DUMP_STARTED]
 [!end]

 write
 write  *** ,, Incremental backup of directory [!dir] on [!date]:[!time] ,, ***
 write

 [!equal,comment,] Get person's name and write to file
 :DUMPERS_NAME.[!end]
 delete/2=ignore  :DUMPERS_NAME
 write/l=:DUMPERS_NAME  [!read  Please type your name: ]

 pop
 dir/i
 [!equal comment,] Post the needed mount and dump commands in batch.[!end]
 qbatch/m/operator
 superuser  on
 dir  :

 [!equal,comment,] Issue a mount command for 10 labeled tape volumes. The
 volids for incremental backup are INCR00, INCR01, etc. (First, to prevent
 problems, delete any linkname of the name MYTAPE in user's directory.)
 [!end]

 delete/2=ignore :udd:[!username]:MYTAPE

 mount/extend/volid=incr00/volid=incr01/volid=incr02/volid=incr03&
/volid=incr04/volid=incr05/volid=incr06/volid=incr07/volid=incr08&
/volid=incr09  MYTAPE   Please mount the longest tape you can.

 [!equal,comment,] Now back up to the labeled tape file. The fileset name
 used here is ROOT. The template does not exclude UDD or any nonmaster
 LDUs. If you want a backup separate from ROOT (like UDD), exclude the
 directory(s) via the dump template. Then insert a "mount/volid=dirnn"
 and "dump ...  dir:#" command after the  dismount command. For example,
 to exclude UDD from the ROOT dump and include it in a separate backup:
 1. Add the text  \UDD:#  to the ROOT dump template.
 2. Add commands  mount/volid=UDD00/vol=UDD01... MYTAPE Please  and
            dump  ... MYTAPE:UDD  UDD:#\+.BRK\+.ED\+.LS
   after the dismount command.
```

Figure 5–3 INC_DUMP.CLI Macro for an Incremental Labeled Tape Dump (continued)

```
  The listing file (/l) in batch is the line printer. [!end]

  write/l  This is an incremental backup of directory [!dir] started at
  write/l  [!date]:[!time] by [DUMPERS_NAME]
  string [LAST_DUMP_DATE]
  write/l  It backs up all files created or modified since [!string]

  dump_ll/v/l/buffersize=8192/retain=0/after/tlm=[!string]  &
     :UDD:[!username]:MYTAPE:ROOT  &
    #\NET\PAGE\PER\PROC\QUEUE\SWAP\SWAP.SWAP\SYSGEN&
\?+.BRK\+.ED\?+JOB\+.LS\?+.TMP  NET:NETGEN:+

  dismount MYTAPE  Incremental backup of directory [!dir] is done.
  pause 5
  send 2 ,, Check the printer — file [!username].OUTPUT.n — for errors.
  send 2 ,, A list of files backed up is printed in file [!username].LIST.n

  [!equal,comment,] Delete and rename file LAST_DUMP_DATE... but do it
  only if file DATE_DUMP_STARTED exists. This prevents LAST_DUMP_DATE
  from being deleted if the batch job was aborted. The terminating
  paren is needed for qbatch/m syntax. [!end]

  [!nequal,([!filenames DATE_DUMP_STARTED]),()]
    delete/2=ignore  LAST_DUMP_DATE
    rename/2=ignore  DATE_DUMP_STARTED  LAST_DUMP_DATE
  [!end]
)
[!else]
    write Error – File LAST_DUMP_DATE doesn't exist! Cannot do incremental
    write backup  without it. Suggest full backup. If you can remember the
    write date of the last backup — full or incremental — create a
    write file named LAST_DUMP_DATE containing this date and retry %0%
    write The form of the date is dd–mmm–yy — for example
    write 09–MAY–85
    pop
[!end]
[!ELSE]
  write This macro is nonexecutable. To make it executable you must use
  write a text editor to change the 2 in line 1 to 1 — both numbers
  write must be 1.
  write
  write Backup is very important. Before using the edited macro
  write for backup you should understand how it works. Please test this
  write macro. Use FULL_DUMP and this macro to back up files. Then try
  write restoring files using the RESTORE_TAPE macro — before relying
  write on this macro routinely.
[!END]
```

Figure 5–3 INC_DUMP.CLI Macro for an Incremental Labeled Tape Dump (concluded)

Full Backup Example

You make sure time–sharing users are logged off and that the CEO, INFOS II, and/or DG/DBMS processes (if you have them) are shut down.

You log on a user console with a privileged username and password. Then type

) FULL_DUMP ↵

Last dump was done on 15–Nov–93:17:56:22

The first time it's run, the macro displays the messages *File does not exist* and *Creating* about the file LAST_DUMP_DATE.

**** Full dump of directory : on 22–Nov–93:18:22:40 ****

Please type your name: JONATHAN ↵

After a short pause, the system console will display

From Pid 0005 : (XMNT) 22–Nov–1993 18:23:40
****
Labeled Mount Request
****
MID = 3448 USER = JONATHAN
User Pid = 49 Requestor Pid = 55
Volumes: FULL00, FULL01, FULL02, FULL03, FULL04, FULL05, FULL06, FULL07, FULL08, FULL09

From Pid 0005 :
Mount Volume: FULL00
Settings: Default Density
User Message: Please mount the longest tape that you can.

Respond: CX MOUNTED @unitname
or CX REFUSED

Write–enable the longest tape your largest tape unit can hold. Mount the tape on a unit, say MTB0. Use the panel switch on the tape unit to select the highest density (if there's a choice). If you've already used the macro, the tape's already labeled and you needn't relabel it.

If you haven't run the macro on this tape, relabel it by typing

) X LABEL @MTB0 FULL00 ↵

Tell EXEC that the tape is mounted.

) CX MOUNTED @MTB0 ↵

The dump begins on this volume. If all specified material fits, the macro issues DISMOUNT and EXEC prompts for a dismount on the system console. But probably, all material won't fit on one volume and at the end–of–tape mark, EXEC will display

```
From Pid 0005 : (XMNT)  22-Nov-1993  18:37:00
                        **********
                        Next Volume Request
                        **********
                        MID =     3453      USER = JONATHAN
                        User Pid =    49     Requestor Pid =   37
                        Volumes:            FULL00, FULL01, FULL02, FULL03,
                                            FULL04, FULL05, FULL06, FULL07,
                                            FULL08, FULL09
                        Units:              @MTB0/FULL00
From Pid 0005 :
                        Mount Volume:       FULL01
                        Settings:           Default Density
                        User Message:       Please mount the longest tape that you
                                            can.
                        Default Unit:       @MTB0

                        Respond:            CX MOUNTED [@unitname]
                                  or        CX REFUSED
```

Remove volume FULL00 from the tape unit, mount volume FULL01 on it, and type CX MOUNTED. (If you have several tape units, you can premount volume FULL01 and FULL02 on other units, saving steps.) If you mount a volume that has the wrong label or fileset ID, EXEC will issue the messages *Mount Error* and *Incorrect labeled volume mounted* and prompt for the correct volume from the MOUNT volume ID list. You can either dismount the current tape and find and mount the correct tape, or use X LABEL to relabel the tape; then type CX MOUNTED. This sequence repeats until the system has dumped all files (except those excluded in the dump template) to the tape fileset.

You will see the following message on the system console:

```
From Pid 0005 : (XMNT)  22-Nov-1993  19:03:00
                        **********
                        Unit Dismount Request
                        **********
                        MID =     3490      USER = JONATHAN
                        User Pid =    49     Requestor Pid =   49
                        Volumes:            FULL09
                        Units:              @MTB0/FULL09
From Pid 0005 :
                        Dismount Unit:      @MTB0
                        User Message:       Full dump of directory : done.

                        Respond:            CX DISMOUNTED [@unitname]
```

You can dismount the tape(s) and type CX DISMOUNTED. Once you dismount the tapes you've completed the full dump. You can file the unused volumes (if any). If you needed extra volumes, note their names on the dump listing and think about adding their volume IDs to the dump macro MOUNT command.

If 10 volumes aren't enough for the full backup, EXEC prompts for another tape. Mount one using the LABEL command to label it if needed. (Continue the same volume ID sequence: FULL11, FULL12, and so on.)

As with any user mount, you can refuse a request (CX REFUSED) to restart. Or you can cancel the batch job by typing QCANCEL and the sequence–number or CX FLUSH and the sequence–number. Canceling the batch job will abort a dump in progress; you will need to type CX DISMOUNTED on the system console.

After the dump finishes, make sure each reel has a paper label with its volume ID, the file set name (ROOT), and the date. Store the reels safely in order. You can use them again for another full dump.

You can reuse the backup tapes as often as desired, for example, every 6 weeks for full dumps, and every 10 working days for incremental dumps. How often you decide to reuse your backup tapes depends on how many backup sets (full and all incremental) you want to keep.

Incremental Dump Example

As with the full dump, you make sure time–sharing users are logged off and that the CEO, INFOS II, and/or DG/DBMS processes (if you have them) are shut down.

You log on a user console with a privileged username and password. Then type

) INC_DUMP ↵

If the file with the last dump date doesn't exist, it will tell you to run a full dump and stop.

Last dump was done on 22–Nov–93:19:03:00

This dump will back up all dumpable files created or modified since 22–Nov–93:19:03:00

**** Incremental dump of directory : on 25–Nov–93:18:10:30 ****

Please type your name: JONATHAN ↵

After a brief pause, the system console will display

From Pid 0005 : (XMNT) 25–Nov–1993 18:12:40
****
Labeled Mount Request
****
MID = 3448 USER = JONATHAN
User Pid = 49 Requestor Pid = 55
Volumes: INCR00, INCR01, INCR02, INCR03, INCR04, INCR05, INCR06, INCR07, INCR08, INCR09

From Pid 0005 :
Mount Volume: INCR00
Settings: Default Density
User Message: Please mount the longest tape that you can.

Respond: CX MOUNTED @unitname
or CX REFUSED

As with the full backup, get the longest tape your largest tape unit can hold. Make sure that the tape is write enabled. Mount it on a unit. Use the switches on the tape unit to select the highest density (if there's a choice). If you've already used the macro, the tape's already labeled and you needn't relabel it. If not run the macro on this tape, relabel the tape by typing

) X LABEL @MTB0 INCR00 ↵

Tell EXEC that the tape is mounted:

) CX MOUNTED @MTB0 ↵

The dump begins on this volume. If all modified files fit, the macro issues a DISMOUNT request and EXEC prompts for a DISMOUNT command on the system console. Often all incremental dump files will fit on one volume. If all modified files won't fit, the system will reach the end–of–tape marker and EXEC will display

```
From Pid 0005 : (XMNT)  25–Nov–1993  18:27:00
                        ********************
                        Next Volume Request
                        ********************
                        MID =     3453      USER = JONATHAN
                        User Pid =    49     Requestor Pid =    37
                        Volumes:            INCR00, INCR01, INCR02, INCR03,
                                            INCR04, INCR05, INCR06, INCR07,
                                            INCR08, INCR09
                        Units:              @MTB0/INCR00
From Pid 0005 :
                        Mount Volume:       INCR01
                        Settings:           Default Density
                        User Message:       Please mount the longest tape that you
                                            can.
                        Default Unit:       @MTB0

                        Respond:            CX MOUNTED [@unitname]
                                  or        CX REFUSED
```

If EXEC prompts for another volume, remove the old one from the unit, mount the next (here, volume INCR01), and type CX MOUNTED. (If you have several units available, you can premount volume INCR01 on another unit, saving a step.) If you mount a volume that has the wrong label or fileset ID, EXEC will say *Mount Error* and *Incorrect labeled volume mounted* and prompt you to mount the correct volume from the MOUNT volume ID list. You can either find and mount the correct tape or type X LABEL to relabel a scratch tape. Then type CX MOUNTED.

This sequence repeats until the system has dumped all files (excluding those excluded in the dump template) to the tape fileset.

```
From Pid 0005 : (XMNT)  25–Nov–1993  19:03:00
                        **********************
                        Unit Dismount Request
                        **********************
                        MID =     3490      USER = JONATHAN
                        User Pid =    49     Requestor Pid =    49
                        Volumes:            FULL09
                        Units:              @MTB0/FULL09
From Pid 0005 :
                        Dismount Unit:      @MTB0
                        User Message:       Incremental dump of directory : done.

                        Respond:            CX DISMOUNTED [@unitname]
```

You can now dismount the tape(s), and type CX DISMOUNTED. Once you dismount the tapes you've completed the incremental dump. You can file the unused volumes. If you needed extra volumes, note their names on the dump listing, and think about adding their volume IDs to the dump macro.

Nearly always, 10 volumes will be enough for an incremental backup. (If not, EXEC will ask for another tape. Mount one, and label it if needed using LABEL and the original volume ID sequence — INCR11, INCR12, and so on. You will see the following message on the system console,

As with any user mount, you can refuse a request by typing CX REFUSED to restart, or you can cancel the batch job by typing QCANCEL and the sequence–number or CX FLUSH and the sequence–number. Canceling the batch job will abort a dump in progress; you will need to type CX DISMOUNTED at the system console.

After the dump finishes, make sure each reel has a paper label with its volume ID, the fileset name (ROOT), and the date. Store the reels safely in order. You can use them again for another incremental dump.

You can reuse the backup tapes as often as desired, for example, every 6 weeks for full dumps, and every 10 working days for incremental dumps. How often you decide to reuse your backup tapes depends on how many backup sets (full and all incremental) you want to keep.

These dump macros, with the restore macro, can save a lot of time, effort, confusion, and data.

Verifying Data Dumped to Tape

Usually, dumped material will load perfectly, restoring your old file structure. This is because on most tape drives, dumped material will load correctly because the tape drives have a read head that verifies material when it is dumped (written by the write head). There is an exception to this general design. Model 6351 tape drives have only one head. For this reason, it is possible, especially with old tape, to dump to a tape that cannot be read.

If you want to make sure that a backup tape is usable, rewind the tape after the dump, and use LOAD_II with the /N (/NLOAD) switch. Run in this way, LOAD_II verifies the dumpfile format and reads every tape record.

For example,

```
) REWIND @MTJ0 ↵
) LOAD_II/N @MTJ0 ↵
... (names of dumped files) ...
)
```

You can verify each tape from the system console, if desired, or after the entire dump finishes.

Restoration Macro for Tape

The macro RESTORE_TAPE.CLI shown in Figure 5–4, restores material from either a full or incremental backup. By default, it uses the volume IDs used for incremental backups, but you can tell it to restore from a full backup. RESTORE_TAPE.CLI requires only the starter system disk structure, as created by a system tape, to restore incremental and full backups.

The RESTORE_TAPE.CLI Macro

```
[!EQUAL,1,2]
[!equal,comment,]
This macro restores files from either a full or an incremental backup
(done by the FULL_DUMP or INC_DUMP macros), in batch. The macro expects
the tape volume IDs to be of the form FULLnn or INCRnn, as created by
the backup macros. The default is INCRnn. The macro accepts pathname
arguments, so you can use it to restore individual files. To work, the
macro requires a basic system file structure, which you can restore
using your system SYSTAPE tape if needed.

The tape fileset name used for the restoration is ROOT. The macro also
explains how to specify a backup to a second fileset (like UDD), if you
have backed up files to filesets other than ROOT.

To execute the macro, the person acting as system operator must go to a
user terminal, log on as a Superuser (like OP), and type the macro name.
Tape mounts and CONTROL @EXEC MOUNTED commands at the system
console will be needed. The most efficient course is to restore the last
incremental backup first, then proceed backwards though most recent full
backup.

The command and pseudomacro syntax used is explained in the manual
Using the CLI.[!end]

[!equal,comment,] Make sure EXEC operator mode is on. [!end]

[!nequal,(ON),([!operator])]
  write Error – Operator is not on. Please go to the system console and
  write type [!read CONTROL @EXEC OPERATOR ON NEW LINE. Return here
  write and press NEW LINE.]
[!end]

write The default restoration assumes you are restoring from incremental
write backup tapes — with volume IDs INCR00 and INCR01 and INCR02 and
write so on. If you want to restore a full backup — with volume IDs
write FULL00 and FULL01 and FULL02 and so on — you must specify FULL.
write
write  For incremental press NEW LINE.
string [!read For full type FULL and press NEW LINE. ]
write
```

Figure 5–4 RESTORE_TAPE.CLI Macro to Restore Dumped Files (continued)

```
delete/2=ignore  :UDD:[!username]:?DUMP_TYPE.TMP
[!equal,([!string]),(FULL)]
  write ,, *** Restoring full backup. Need volume IDs FULL01 FULL01 etc. ***
  write/l=:UDD:[!username]:?DUMP_TYPE.TMP  FULL

[!else]
  write  *** Restoring incremental backup. Need volume IDs INCR01
  write  INCR01 etc. ***
  write/l=:UDD:[!username]:?DUMP_TYPE.TMP  INCR
[!end]

[!equal,comment,] Get person's name and write to file :RESTORERS_NAME
[!end]
push
superuser on
dir  :
delete/2=ignore   :RESTORERS_NAME
write
write/l=:RESTORERS_NAME    [!read  Please type your name: ]
pop
dir/i

[!equal comment,] Post the needed mount and dump commands in batch.
[!end]
qbatch/m/operator
superuser  on
dir  :

[!equal,comment,] Issue a mount command for 10 labeled tape volumes.
The volids depend on the choice above — INCRnn or FULLnn. (First, to
prevent problems, delete any linkname of the name MYTAPE in the user's
directory.) [!end]

delete/2=ignore   :udd:[!username]:MYTAPE
string  [:UDD:[!username]:?DUMP_TYPE.TMP]

mount/extend/volid=[!string]00/volid=[!string]01/volid=[!string]02&
/volid=[!string]03/volid=[!string]04/volid=[!string]05/volid=[!string]06&
/volid=[!string]07/volid=[!string]08/volid=[!string]09   MYTAPE   &
Please mount the correct tape volume.

[!equal,comment,] Now load from the labeled tape file. If the person
included pathname arguments 1–n, include them in the load command line.
If he/she omitted arguments, restore the entire backup.
```

Figure 5–4 RESTORE_TAPE.CLI Macro to Restore Dumped Files (continued)

```
The fileset name used here is ROOT. It does not include any directory
that was excluded from ROOT in the backup macros. If you have a backup
fileset separate from ROOT, you must load it separately, by fileset
name. Use the syntax shown in the mount and load commands above, with
your own custom volid names and fileset name. If the backup file
includes any nonmaster LDUs, be sure they're initialized before you load
from the backup tape. (Otherwise, the load command will recreate the
dump file directory on the master LDU.)

The listing file (/l) in batch is the line printer. [!end]

write/l  This is an [!string] restoration of directory [!dir] started
write/l  at [!date]:[!time] by [RESTORERS_NAME]

load_ii/v/l/buffersize=8192/recent        &
   :udd:[!username]:MYTAPE:ROOT   &
   %1-%

dismount MYTAPE  [!string] restoration of directory [!dir] is done.
pause 5
send 2 ,, Check the printer — file [!username].OUTPUT.n — for errors.
send 2 ,, A list of files restored is printed in file [!username].LIST.n
send 2 ,, Don't forget to restore the other backups, if this applies.
delete/2=ignore  :UDD:[!username]:?DUMP_TYPE.TMP

[!equal,comment,]  The following paren is needed for qbatch/m syntax.
[!end]
)
[!ELSE]
 write This macro is nonexecutable. To make it executable you must use
 write a text editor to change the 2 in line 1 to 1 — both numbers
 write must be 1.
 write
 write Backup is very important. Before using the edited macro
 write for backup you should understand how it works. Please test this
 write macro. Back up some files using FULL_DUMP and INC_DUMP. Then
 write restore files using this macro before relying on the macro set
 write routinely.
[!END]
```

Figure 5–4 RESTORE_TAPE.CLI Macro to Restore Dumped Files (concluded)

Restoring Files from Backup Tapes

Restoring falls into two categories: restoring one or more files, and restoring one or more LDUs. The first category is more common, easier, and faster.

Restoring One or More Files

Usually people restore one or more files when someone has accidentally deleted a file (perhaps a directory) or group of files. Perhaps someone was careless with the DELETE command and a template character. If you want to restore files that were backed up to tape, there are two things to consider: the tape set(s) needed, and the pathname template.

The tape set(s) you use for restoration depends on the date that the lost file(s) was last modified. If the file(s) was created since the last backup, then it wasn't backed up, and cannot be restored. Otherwise, use the backup that occurred soonest after the file(s) was modified (incremental or full).

If you can't determine when the file(s) was last modified, check the backup listings. If the name of a lost file appears in any listing, then you know the file is in that backup. In the worst case, without a listing or dates, you must restore the last full backup set; then the earliest incremental backup; then the next incremental backup, and so on. This is a good reason to keep your backup listings — especially for incremental backups.

After deciding on the tape set, you must choose one or more pathname templates (unless you want to restore the entire backup). The RESTORE macro allows template arguments in which you can specify a directory, a specific file, or a directory and pathname template.

You can restore all files in and below a directory (including subordinate directories) with the template

pathname–from–root:

You can restore all files in and below a user's directory with the template

UDD:username:# (omit the leading : from UDD)

For CEO files, the directory structure and restoration procedure differs from that for standard AOS/VS or AOS/VS II files. To restore CEO files, see the manual *Managing the CEO® System*.

File Restoration Example

As an example, assume Andy accidentally deleted two files named REPORT.MAY and SUMMARY. He last modified them a week ago. (If lost files were last modified on different dates, it may be most efficient to use the last full backup to restore the lost files). In the role of system operator, you retrieve the appropriate set of tapes.

You need a template for the restoration. The two filenames have the letters MA in common. So you could use the template

UDD:ANDY:#:+MA+

This template would work, but it might restore many matching, unwanted files. You can be more specific. The two names each have at least one character, and no period, following the A. So you could refine the template to

UDD:ANDY:#:+MA*–

If you know which directory the files were in, you could use the specific pathnames,

pathname–from–root:REPORT.MAY pathname–from–root:SUMMARY

Let's say you decide on the most general course — the +MA+ template — to cover many possibilities.

You log on a user terminal with a privileged username and password. Then type

) RESTORE_TAPE UDD:ANDY:#:+MA+ ↵	Start the macro, specifying the desired template.
The default restoration assumes you are ...	The restoration macro describes the two kinds of restorations.
For incremental press new line. *For full type FULL new line.*	Press NEW LINE to select an incremental restoration.

Restoring incremental backup — expect volume IDs INCR00...

Please type your name: JONATHAN ↵

... (pause) ...

After a brief pause, the system console will display

From Pid 0005 : (XMNT) 25–Nov–1993 18:12:40
****
Labeled Mount Request
****
MID = 3448 USER = JONATHAN
User Pid = 49 Requestor Pid = 55
Volumes: INCR00, INCR01, INCR02, INCR03,
INCR04, INCR05, INCR06, INCR07,
INCR08, INCR09

From Pid 0005 :

Mount Volume:		*INCR00*
Settings:		*Default Density*
User Message:		*Please mount the correct tape volume.*
Respond:		*CX MOUNTED @unitname*
	or	*CX REFUSED*

Mount the first tape in the fileset on a unit, say MTB0.

Tell EXEC that the tape is mounted:

) CX MOUNTED @MTB0 ↵

The restoration begins from this volume. If this is the only volume in the tape set, or if the system has restored all files you specified (you must have specified entire filenames, not templates), then the macro issues DISMOUNT and EXEC prompts to dismount the tape on the system console. For the sake of this example, assume another volume is needed. At the end–of–tape mark on volume INCR00, EXEC displays

From Pid 0005 : (XMNT) 25–Nov–1993 18:27:00
****
Next Volume Request
****

MID =	*3453*	*USER = JONATHAN*
User Pid =	*49*	*Requestor Pid = 37*
Volumes:		*INCR00, INCR01, INCR02, INCR03, INCR04, INCR05, INCR06, INCR07, INCR08, INCR09*
Units:		*@MTB0 / INCR00*

From Pid 0005 :

Mount Volume:		*INCR01*
Settings:		*Default Density*
User Message:		*Please mount the longest tape that you can.*
Default Unit:		*@MTB0*
Respond:		*CX MOUNTED [@unitname]*
	or	*CX REFUSED*

Remove volume INCR00 from the tape unit, mount volume INCR01 on it, and type CX MOUNTED and press NEW LINE. (If you have several tape units, you can premount volume INCR02 on another unit, saving a step.) If you mount a volume that has the wrong label or file set ID, EXEC will display *Mount Error* and *Incorrect Labeled volume mounted*, and prompt you to mount the correct volume in the MOUNT volume ID list. Dismount the tape, find and mount the correct tape; then type CX MOUNTED and press NEW LINE.

This sequence repeats until the system has restored the files or read the entire fileset of tapes. When you see the following message on the system console,

From Pid 0005 : (XMNT) *25–Nov–1993 19:03:00*
****
Unit Dismount Request
****
MID = 3490 USER = JONATHAN
User Pid = 49 Requestor Pid = 49
Volumes: INCR09
Units: @MTB0 / FULL09
From Pid 0005 :
Dismount Unit: @MTB0
User Message: INCR restoration of directory : is done.

Respond: CX DISMOUNTED [@unitname]

You can dismount the tape(s) and type CX DISMOUNTED. Once you dismount the tape(s) you are finished with the restore. Check the printer for verification of files restored; and store the tape reels safely.

As with any user mount, you can refuse a request by typing CX REFUSED to restart; or you can cancel the batch job by using the format

QCANCEL sequence–number

or

CX FLUSH sequence–number

Canceling the batch job will abort a restoration in progress, you will need to type CX DISMOUNTED on the system console to tell EXEC you are dismounting a tape.

With the files restored, tell Andy to check them and to check his directories for unwanted, restored files. He can then delete the unwanted files.

(If Andy had created the files in the CEO system and had accidentally deleted the documents named REPORT.MAY and SUMMARY, suggest that he check the CEO wastebasket. In the full CEO system, documents are not actually deleted until a program called Janitor has been run. Until then, documents that people delete can be retrieved from their wastebaskets. If Andy's documents are not in his wastebasket, you can restore them from the last backup as described in the manual *Managing the CEO® System*.

Shortening a Restoration

When you mention a specific filename (like UDD:SAM:MYFILE) in a restoration, the restoration will end and the CLI prompt will return as soon as the file has been copied to disk.

The restoration will take longer if you use a template like UDD:SAM:#:MYF+ because the system will continue through the last tape, even after the desired file has been copied. The system can't tell, until it reaches the end of the last tape, that there is no matching filename in the tape fileset.

Generally, it's desirable to give a specific filename if you can. If you do, be sure it's the correct one. If you make a mistake with a specific filename, the system will take you all the way through the tape set, and not restore the file you want restored. If you think you made a mistake you can always cancel the batch job by typing QCANCEL and the job sequence number, and pressing NEW LINE. (The batch job sequence number is displayed by the QDISPLAY command). Then start again.

Restoring an Entire LDU Using Backup Tapes

The time may come when you need to restore an entire LDU from backup tapes. The data on an LDU may need to be replaced if a disk wears or fails in such a way that the system can no longer read it.

If you have checked with Data General, and either acquired a new disk or decided to rebuild the old one, follow these steps:

To restore the master LDU, continue. To restore a nonmaster LDU, skip to step 5.

1. Get the tailored system tape you made with SYSTAPE after testing your tailored AOS/VS or AOS/VS II system. (If you don't have a tailored system tape, use the latest system tape you received from DG; later, you'll need to generate a tailored system.)

2. Turn to the chapter in the *Installing* manual for your particular operating system that explains how to bring up your computer system and execute all the numbered steps there, using your tailored system tape or the DG tape. (If you're not using a tailored system tape, generate a tailored operating system, patch it, start it, and make a system tape — all as described in the *Installing* manual for your operating system.

3. If you have any DG software products that were not backed up (those under :UTIL, for example), install these as described in the software manuals for those products.

4. Use PREDITOR to create an operator profile; then bring up EXEC and create line printer/batch queues (as shown in the *Installing* manual for your particular operating system.

5. To restore a nonmaster LDU, make sure this LDU is formatted with either the Disk Formatter (AOS/VS) or Disk Jockey (AOS/VS II) and has the desired name. Then initialize it into your system as usual.

   If the nonmaster LDU needs DG software installed (for example, FORTRAN 77), and that software was not backed up, make that LDU the working directory and install the software from the release media.

Follow this step for all nonmaster LDUs you want to restore.

6. Get all your incremental backup sets and your last full backup set of tapes.

7. Log on a user terminal that's physically close to the system console. Restore your most recently dumped incremental backup tape set. Move backwards chronologically through the incremental backups until you have restored them all. Then restore the most recent full backup.

   After each restoration ends, check the printed batch output file for error messages. If you see a *Control point directory max size exceeded* message, the LDU is full. You (or other users) must delete some previously deleted (but restored) files to free up some space, before you can continue the restoration process.

   The files to delete may be in user directories, where they can be deleted with the CLI DELETE command or in the CEO system, where they can be deleted with option 5, "Delete," on the CEO Documents screen. In the full CEO system, you will also need to run the Janitor to complete the deletions.

   When you have some disk space free, 1000–5000 blocks or more (type SPACE : and press NEW LINE to find out how much free space is available), proceed with the next restoration.

8. Make sure you replace tape covers and store the tapes safely.

9. You're done! You've recreated the entire LDU(s) — with luck, losing only a little work (the files created or changes made since the last backup occurred).

   If you have XODIAC™ networking software, the :NET:NETGEN files were backed up and have been restored; you can regenerate host and RMA files by running NETGEN and specifying the network specification file filename.

The RESTORE macro restores files with their original creation times, but it changes the time last modified to the time you restore the files. This means that your next backup must be a full backup, via the FULL_DUMP macro, since the dates last modified aren't real. It also means that the FILESTATUS command with the /BEFORE/TLM= switches won't help identify old files after the restoration.

Hard Tape Error Recovery with the DUMP_II and LOAD_II Utilities

Unlike the DUMP and LOAD commands, the DUMP_II/LOAD_II utilities can recover from hard tape errors. This section describes how DUMP_II and LOAD_II handle errors.

The DUMP_II Utility's Error Recovery Capability

When a hard tape error occurs while backing up files, DUMP_II could write an error message like the following on your terminal's screen:

Physical unit failure
Warning: Too many tape retries, file @MTB1
Hard tape error, block: n
The tape is rewinding ...
Mount the next volume, volume: n
Respond MOUNTED <device> or REFUSED when ready.

It is wise at this point to discard the tape and restart the backup. If you are pressed for time, however, you may want to mount the next unlabeled volume and respond to the prompt by typing MOUNTED, the device name (for example, @MTB1), and pressing NEW LINE. (You could also change devices if you suspect problems with this tape drive.)

CAUTION: *In rare cases, MTJ–type tape drives except for the 21–Mbyte cartridge tape may generate the message* Fatal buffered tape error *during a dump.*

These drives cannot recover from this kind of error. If you get this error, and you know that the problem is not with the drive, discard the tape and do not reuse it. Restart the dump. See the Notes and Warnings section of the release or update notice for the latest status of this problem.

The LOAD_II Utility's Error Recovery Capability

When a hard tape error occurs while loading files, LOAD_II could write an error message like the following on your terminal's screen:

Warning: Too many tape retries, file @MTB1
Hard tape error, block: n
Block n skipped.
Indecipherable dump format, file <filename1>

File is inconsistent after byte n
Searching for the next file ...
n bytes skipped.

Does file <filename2>
belong in current directory <directoryname>? (Yes/No/Quit): Y ↵

Here, LOAD_II can't read a portion of the tape. It notifies you that it is searching for the next good block on the tape, and — not knowing whether your current working directory would have changed had it been able to read the intervening blocks — asks whether the good file belongs in the current working directory.

Labeled Tapes

Although you can use the DUMP_II/LOAD_II utilities to make backups — even multiple volume backups — without using labeled tapes, there are good reasons to use labels. Labels provide a measure of protection from operator errors by providing information that you can check as you mount each volume. When you load or dump multiple volume backups, labels ensure that you mount the volumes in the correct order. In addition, the label provides some measure of documentation for the information stored on the tape.

A labeled tape begins with a block of information that identifies the contents of the tape. Among other things, the label specifies a volume identifier (volid), which uniquely identifies the tape. (See Figure 5–1.) You can specify several volids for any single dump or load operation. Thus, a dump file can span several physical volumes of tape.

Labels created by both the CLI DUMP command and the DUMP_II utility are compatible with the American National Standard for Magnetic Tape Labels (ANSI x3.27–1978) Level 3.

To ensure error handling capability, you should always use the DUMP_II and LOAD_II utilities, rather than the CLI commands DUMP and LOAD.

DUMP_II and LOAD_II work equally well with labeled and unlabeled tape(s).

A discussion of labeled tapes and how to use them can be found in Chapter 3 of this manual. The details of specific switches pertaining to labeled tape volumes are found in the descriptions of the DUMP/LOAD commands and DUMP_II/LOAD_II utilities found in the *Using the CLI (AOS/VS and AOS/VS II)* manual.

Error Handling Using Labeled Tape

When you dump and load using labeled tapes, there are many error conditions that you can encounter. These errors fall into three categories: user errors, operator errors, or media errors. An example of a user error is requesting the wrong tape volume; an example of an operator error is mounting the wrong tape volume; and, an example of a media error is a hard tape error.

DUMP_II and LOAD_II assume that the operator's job is to mount the volume requested. If the operator mounts a volume whose volid does not match the volume requested, DUMP_II/LOAD_II continues to request the volume until the operator mounts the correct volume or refuses the request.

Any other errors are referred to you, as the user, so that you can decide what to do. Even if the operator refuses a mount request, you are notified and given an opportunity to decide what to do.

The various error conditions referred to in the following paragraphs are described in detail in the manual *AOS/VS and AOS/VS II Error and Status Messages*.

Errors Reported by LOAD_II

You must be sure an operator is available to mount and dismount labeled tapes. (An operator must issue the CONTROL @EXEC OPERATOR ON command.) If there is no operator, and you issue an implicit mount request, LOAD_II will terminate with the following message:

Operator not on duty — restart after doing "cx operator on"

If the operator refuses the mount request, then you will receive the following error:

Warning: Refused by operator

If the operator refuses a request for a volume, you are prompted to retry the request, specify another tape volume, or quit.

The *Retry, New volume, or Quit* prompt allows you the option of continuing with a dump or load operation after telling the operator what should be done about the request.

Labeled Tape Errors Sent to the User

When you specify a labeled tape file in the command line, the LOAD_II utility checks for various problems and reports any error conditions that exist. The error messages you see when specifying labeled tape are:

VOLID is missing from labeled media specification
VOLID must be 6 characters or less
VOLID contains illegal characters
File Identifier is missing from labeled media specification
File Identifier must be 17 characters or less
File Identifier contains illegal characters
Labeled media specification should look like: @LMT:VOLID:FILEID

After finding and reporting any of these errors, LOAD_II terminates. Correct the error and retry the request.

Labeled Tape Errors Sent to the Operator

The LOAD_II utility reports other kinds of errors with labeled tape to the system operator, including:

Unreadable tape label
Incorrect labeled volume mounted

The message *Unreadable tape label* means that the operator mounted an unlabeled rather than a labeled tape. If this occurs on a DUMP_II request, the operator should label the tape and remount it. If this mount request is part of a load operation, the operator should check to see that he or she mounted the proper tape volume, or refuse the mount request.

The message *Incorrect labeled volume mounted* means that the operator mounted the wrong volume of a tape set. The operator should check the volume and either mount the correct volume or refuse the mount request.

Errors in Specifying Start of Volume or File Header Label

Errors reported to the user concerning the Start of Volume and File Header Label include the following:

Volume mounted is empty – contains no data
Label contains incorrect owner, expected: <name>, found: <name1>
Tape is not correctly labeled – HDR1 label not found
Label contains incorrect fileset name, expected: <name>, found: <name1>
Tape not correctly labeled – HDR1 label contains no creation date
Label contains incorrect creation date, expected: <date>, found: <date1>
Label contains incorrect filename, expected: <name>, found: <name1>
Label contains incorrect file sequence number, expected: <num>, found: <num1>
Label contains incorrect file section number, expected: <num>, found: <num1>
Label contains incorrect expiration date, expected: <date>, found: <date1>
Labeled tape contains unlabeled file, tape file number: <num>

The prompt for any of these errors is the same:

Specify new volume, *Ignore label*, or *Quit?*

If you want to select a new tape volume, respond with the volume identifier (volid) of the new volume. EXEC will then prompt the operator to mount the new volume. The volid you supply with the *Specify new volume* request may be different than or the same as the volid on your original request. If you type an incorrect volid when you request a tape, respecify the correct volid. If the operator mounts a volume with the correct volid, but from another tape set, respecify the same volid.

If you select *Ignore label*, and loading has not started, you will receive the following prompt:

Continue searching for tape filename <name>, or Load next data file? (C,L):

If you select *C,* the LOAD_II utility skips the current file and continues looking for the specified filename. If you choose *L*, LOAD_II ignores the error in the tape label and loads the next physical file.

If a load operation has started, one of the above errors occurs and you choose *Ignore label*, the load proceeds with the next physical file. No additional response is required.

If LOAD_II utility encounters a hard tape error while reading a File Header Label, the utility displays the following message:

HDR1 label contents lost due to media failure

If this error occurs while the LOAD_II utility is searching for the correct file, LOAD_II will display the following prompt:

Continue searching for tape filename <name>, or Load next data file? (C,L):

As explained above, if you want LOAD_II to skip over the current file and continue searching for the filename, choose *C*. If you want the utilities to ignore the HDR1 label and load the next physical file, choose *L*.

If the LOAD_II utility finds the file in question, and this error occurs when LOAD_II is processing labels on subsequent volumes, the following prompt appears:

Continue loading data, or Quit?

If you choose to continue, LOAD_II ignores the bad label and proceeds. If you choose to continue loading data after the first prompt, LOAD_II assumes that the file to be loaded has an unreadable label and proceeds to load data.

Errors in Specifying End–of–File or End–of–Volume Labels

If LOAD_II encounters an error while processing an end–of–file or end–of–volume label, the utility will display one of the following error messages:

Warning: <label type> Label contents lost due to media failure
Missing EOF label after file: <num>, while searching for tape file
Missing EOV label while searching for tape file

The prompt that you see is

Continue searching or Quit?

If you choose to continue, LOAD_II skips over the bad label and continues the search.

If LOAD_II finds an unlabeled tape file while looking for a file in a fileset, the utility sends the following message:

Labeled tape contains unlabeled file, tape file number: <num>

Finding an unlabeled file on a labeled tape can occur if an unlabeled dump operation has been done to a specific file on a labeled tape.

The LOAD_II utility will prompt you with the following message:

Continue searching or Quit?

Errors Reported by DUMP_II

When the DUMP_II utility searches for an expired file or the last file on a volume, the labels are processed in much the same way as for a load operation. The volume that the operator mounts must have the correct VOLID and if this dump operation will use more than one tape volume, DUMP_II checks to see that the volumes are mounted in the correct order.

If DUMP_II encounters any problems reading an existing tape set while trying to find where to start the dump, the only options open to you are to specify a new volume or quit. DUMP_II does not allow you to continue the dump operation, because using a tape with any kind of hard tape errors only perpetuates the problem. When you encounter hard tape errors on a tape during a dump operation, mount a new tape and begin again.

The errors that the DUMP_II utility reports while searching for where to start the dump are:

Label contains incorrect owner, expected: <name>, found: <name1>
Tape is not correctly labeled – HDR1 label not found
Label contains incorrect fileset name, expected: <name>, found: <name1>
Tape not correctly labeled – HDR1 label contains no creation date
Label contains incorrect creation date, expected: <date>, found: <date1>
Label contains incorrect filename, expected: <name>, found: <name1>
Label contains incorrect file sequence number, expected:<num>, found: <num>
Label contains incorrect file section number, expected: <num>, found: <num>
Label contains incorrect expiration date, expected: <date>, found: <date1>
Labeled tape contains unlabeled file, tape file number: <num>
Warning: <label type> label contents lost due to media failure
Missing EOF label after file: <num>, while searching for expired file
Missing EOV label while searching for expired file

The prompt after any of the errors reported by DUMP_II is the same:

Specify new volume, or *Quit?*.

If you choose to specify a new volume type S and press NEW LINE. The DUMP_II utility will respond with the following prompt:

VOLID of new volume?

You can then respond by typing the volume identifier of the new volume. After you specify the new volid, the operator will be prompted to mount that volume.

Backing Up to Diskettes with the DUMP and LOAD Commands

This section gives some pointers on using diskettes for backup: handling diskettes, using DUMP with diskettes, and some macros to help you do it. You must be running the 16–bit CLI in order to execute the DUMP and LOAD commands on diskettes; the 32–bit CLI does not support labeled diskettes.

Backup Sets of Diskettes

One hard disk can hold 120 Mbytes (about 230,000 disk blocks); a diskette can hold 720 Kbytes (about 1400 disk blocks). Thus over a hundred diskettes are needed to back up the biggest disk available on a diskette–only system. For smaller disks (70 or 38 Mbytes), you need proportionately fewer diskettes. At the beginning, you can get by with very few diskettes because your disk will not have filled up. To get the approximate number of diskettes needed, type SPACE : and NEW LINE; divide the CUR figure by 1400.

Assume 3 – 10 diskettes for each incremental backup, and multiply this number by the number of incremental backups you plan between full backups. Add the total incremental backup number to the full backup number. The result gives the approximate number of diskettes needed for one backup set.

You can get along with one backup set, but ideally you should have two or more sets. With two sets, you can keep the last backup set intact and use the previous backup set for the new backup.

Whatever schedule and plan you come up with, be sure you have enough diskettes on hand when you start a full backup. If you run out of diskettes and can't complete a FULL_BACKUP normally, the backup will be incomplete. To make it complete, you'll need to start again from the beginning with enough diskettes. (This caution also applies to incremental backups, but with less force, since only a few minutes are wasted if you need to restart.)

Handling and Storing Diskettes

Diskettes are important and fragile. Some handling cautions and hints follow.

- Store diskettes in their outer envelopes; remove a diskette from its outer envelope just before you use it.
- Hold a diskette by the edges of the envelope only. Avoid touching the diskette surface (exposed in oval cutout on the inner envelope). The oil on your finger could make that part of the diskette unreadable.
- If a diskette has no paper label, apply one. Properly labeled diskettes make life easier. Labels go on the smooth, seamless side of the inner envelope, at the top (with the write–enable notch to the right). Avoid the oval cutout. Remove the sticky–backed label from its backing and apply it to the inner envelope.
- To write on a diskette label, use only a felt–tipped pen. A pencil or ball–point pen can score the diskette surface destroying some or all data on it.
- A diskette must be properly inserted; the operating system can't read one that's improperly inserted. When you insert a diskette, hold it by the edge with label

(seamless) side facing right and the write–enable notch up. The diskette should slide in smoothly and come to a firm stop.

- Diskettes wear. If you see the message

  Soft error, device 64 n

  this is a warning. Consider substituting a new diskette for the one in unit *n*.

  A *Hard Error* or *Physical unit failure* message means that the rest of the diskette is unreadable or unwritable if you are using the DUMP or LOAD commands. If this message appears during backup, replace the diskette. If it appears during a restoration, restart the restoration. If the hard error recurs, this probably means you cannot restore data from the rest of the diskettes in this set. (Perhaps there is another, earlier set you can use.)

  The DUMP_II/LOAD_II utilities provide you with the capability to recover from hard tape errors. DUMP_II/LOAD_II allow you to either retry reading from or writing to the block that caused the error, advancing to the next good block, or terminating the dump or load operation.

- Don't bend or twist a diskette. A crease on the surface means data loss.
- Cold and heat can harm diskettes. Keep them temperate (between 50 and 125 degrees Fahrenheit (10 and 50 degrees Centigrade).
- A magnetic field can erase part or all data on a diskette. Keep diskettes away from electric motors, magnets, and transformers.
- Don't turn computer power off while a diskette is inserted; remove diskettes first. Turning off power while a diskette is inserted can lead to data loss.
- Diskettes must be hardware formatted (different from software formatting, which is done by the Disk Formatter or the Disk Jockey utilities).

  A diskette that is not hardware formatted will produce a *Hard error* or *Physical unit failure* message. To hardware format a diskette, run the hardware formatting program described in the hardware documentation. It's a good idea to hardware format non–DG diskettes immediately after you purchase them.

- Generally, you should not write–protect diskettes. The operating system can't access a write–protected diskette as a directory, or write to it in any way.

Backing Up to Labeled Diskettes

The CLI DUMP and LOAD commands (as well as the FILESTATUS command) accept templates. (The system matches templates, dumping or loading files in the order it finds them, not necessarily in the order you expect.) The DUMP command, without a directory name template, copies all files and directories in the working directory to a dump file. It writes a header before each file, with the file's name, date of creation, ACL, and other data. The directory structure is maintained; later if you load the dump file into the directory from which DUMP was issued, the system will try to recreate the original structure. So it's very important that the working directory be the same for LOAD as it was for DUMP. The root directory (:) is the best directory from which to start backups and restorations.

The CLI can read, write, and label diskettes allowing you to dump and load material using a sequence of labeled diskettes. This kind of sequence, with multiple diskette volumes, is essential for backup if your system doesn't have a tape unit. Labeled diskettes are needed in any situation where someone wants to write to diskette and the material involved won't fit on one diskette.

The command that enables the CLI to access labeled diskettes is OPERATOR. It is available to any process, on any console. The CLI command OPERATOR is very different from EXEC's command of the same name (described in Chapter 3).

The CLI OPERATOR command has the form

OPERATOR [*ON* / *OFF*]

If you omit arguments, the CLI displays the current status of Operator mode. If you include ON, the CLI turns on Operator mode (if it is not already on). If you include OFF, the CLI turns off Operator mode (if it is on).

When Operator mode is on, the CLI can dump to, label, and load from labeled diskettes. Operator mode can be turned on by any CLI process, on any console. It stays on until turned off or until the CLI process terminates.

When Operator mode is off, the CLI cannot access a diskette by label. An attempt to do so provokes a *No operator available* error message. The CLI can, as usual, access a diskette by physical unit name, for example, via the DUMP or LOAD commands, as in

```
) DUMP/V @DPJ10 MYFILE ↵
```

The OPERATOR /LABEL Switch

The /LABEL switch tells the CLI to label diskettes during a dump operation without telling you before doing so.

On any dump to a labeled diskette, the CLI checks the diskette before starting to write. If the diskette has the volume ID (volid) expected, the CLI starts writing to it.

If the diskette does not have the volid expected, the CLI's action depends on whether you included the /LABEL switch when you turned operator on. If you omitted /LABEL, the CLI displays an error message and asks if you want it to relabel the diskette. If you included /LABEL (for example, OPERATOR/LABEL ON), the CLI relabels the diskette without asking for your okay.

The /LABEL switch is useful for dumps when the diskettes you want to use are not labeled, or when you don't care about their labels. It eliminates the label specification step on a label conflict.

Whether or not you include /LABEL, the CLI will create volids for you as described in the next section, under "volid."

Labeled Diskette Access

After you turn Operator mode on (with or without the /LABEL switch), you can use labeled diskettes. To access a labeled diskette, use the form

command @LFD:volid:filename

where:

command — is the DUMP command (to write to) or the LOAD command (to read from) one or more labeled diskettes.

On a dump, if you omitted /LABEL from the OPERATOR command, the CLI checks to see if the retention period specified in the previous dump has elapsed. If it has not elapsed, the CLI asks if you want to relabel the diskette. The default retention period is 90 days. With the DUMP/RETAIN= switch, you can specify a different period, including 0 days (which allows the diskette set to be reused immediately).

@LFD — is the generic filename that indicates a labeled diskette (labeled floppy disk). You must include @LFD for labeled diskette access.

volid — is the volume ID that's on the first diskette (on a load operation) or volume ID you want written on the first diskette (on a dump operation). A volid can be written to a diskette either by the CLI or by the LABEL utility. The CLI's labeling mechanism is more convenient. Volids are limited to six legal filename characters.

On a load, the volid you specify must match the volid on the first diskette. If not, the CLI will signal an error when it checks the diskette label.

On a dump, the CLI checks the diskette label. If the label is correct, the dump proceeds. If the label is wrong, the CLI's message depends on whether you included the /LABEL switch when you turned Operator mode on. If you omitted /LABEL, the CLI warns you that the label doesn't match the one expected. Then it gives you the choice of relabeling the diskette or inserting another diskette. If you included /LABEL, the CLI relabels the diskette without comment.

For access to the second and subsequent diskettes, the CLI creates default volids, based on the original volid you specified. It creates a numeric sequence by adding 1 to each volid. If the volid you originally specified doesn't end with a number, the CLI will add a number, space permitting. If there isn't room for the CLI to add a number, it will drop as many characters as needed to create the next volid.

For example,

Volid You Specify	Default Volids Created by CLI
VOL1	VOL1 (for first diskette) VOL2 (for second diskette, if needed) . . . VOL9 (for ninth diskette, if needed) VOL10 (for tenth diskette, if needed)
SYSTEM	SYSTEM (for first diskette) SYSTE1 (for second diskette,if needed) . . . SYSTE9 (for ninth diskette, if needed) SYST10 (for tenth diskette, if needed)

When you choose a volid, we suggest that you end it with a number, for consistency with the CLI's label–creating sequence (VOL1, VOL2, VOL3, and so on).

filename is the filename of the diskette fileset (LOAD_II); or it is the filename you want to create for the diskette fileset (DUMP_II).

This filename applies to the entire fileset of one or more diskettes. The same filename used for a labeled diskette dump must be used to load the diskette set; if not, the CLI will report a *File does not exist* message when it checks the label of the first diskette. The filename is limited to 17 legal filename characters.

With Operator mode on, after you type a command of the form @LFD:volid:filename, the CLI will prompt you to insert a diskette in a specific unit, like this:

Please insert a diskette if not already inserted
unit [@DPJ10] volume ID [volid] ? [Y]

The default diskette unit is @DPJ10. The displayed volid is the one you specified (first diskette) or the next sequential number after the preceding volid. If you want to override the default unit, type N and press NEW LINE. The CLI then allows you to specify a different default unit.

The volid and filename you specify cannot be changed during a dump or load operation. To use a different volid or filename, you must interrupt the command (using CTRL–C CTRL–A) and restart it. For example, assume you type

```
) DUMP/V @LFD:XVOL1:MYFILE ↵
```

The CLI displays the *Please insert* message shown above. If you decide that you prefer VOL1 to XVOL1 as a first volid, abort the dump using CTRL–C CTRL–A; then retype the command with the new volid. In this case, you'd type

```
) DUMP/V @LFD:VOL1:MYFILE ↵
```

For any dump, each diskette must be hardware formatted, but need not be software formatted with the Disk Formatter. During any load operation, the first diskette must have the volid you specify, and each of the following diskettes must have the volid that the CLI expects. If you make a mistake, like typing the wrong volid or filename, abort the command via CTRL–C CTRL–A and restart it, as explained above.

Diskette Access Control

The default ACL for any diskette unit does not allow user access. If you want users without Superuser privilege to be able to use diskettes, you should set the unit ACL to allow this access. The best place to do this is in the system UP macro, with a command like

```
ACL @DPJ10 +,WARE
```

While you are loading from (or dumping to) diskette, the system does not automatically protect your diskette from access by other users. For example, user Joan can write to a diskette that you (OP) are trying to read. With Superuser on, or if you have owner (O) access to the diskette unit, you can change the unit ACL to prevent other users from accessing it. For example, you might type

Su) ACL/V @DPJ10 ↵ *DPJ10 +,WARE*	The operating system displays the ACL.
Su) ACL @DPJ10 OP,OWARE ↵	Set ACL for your use only.

Then use the unit. When you're finished with the unit, restore its original ACL so others can use it.

Labeled Diskette Example

The following example shows a full backup of the entire system. CLI macros to make such backups easier are shown in the next section. We're including a full backup example, without macros, here to show how it works.

In this example, the diskettes have not been used for a system backup. They have not been labeled and the person doing the backup chooses to include the /LABEL switch for labeling without the extra step of typing the new labels.

.	Ensure that all users have logged
.	off and that all server processes.
.	like CEO and INFOS II, if present,
	are shut down.
) DIR : ↵	Get to the root directory.
) OPERATOR/LABEL ON ↵	Turn Operator mode on, with the /LABEL switch.
) SUPERUSER ON ↵	Turn Superuser on for file access.
Su) ACL @DPJ10 OP,OWARE ↵	Give yourself sole access to the diskette unit.

Su) DUMP/V/L=DFILE @LFD:FULL01:BACKUP ↵

Please insert a diskette if not inserted	The CLI prompts you to insert a diskette.
Unit [@DPJ10] volume ID [FULL01]? [Y] ↵	You insert a diskette and press NEW LINE.
	Since the /LABEL switch is included, the CLI labels the diskette without asking for confirmation. The dump proceeds with filenames listed to file DFILE. Each diskette takes about 2 minutes to fill.
Please insert next diskette *Unit [@DPJ10] volume ID [FULL02]? [Y]*	After the diskette has been filled, the CLI prompts for the next one, incrementing the volid. Remove the diskette, insert the next one and press NEW LINE.
	Again the CLI labels the diskette and the system continues dumping files. The dump proceeds through other diskettes.

Please insert next diskette
Unit [@DPJ10] volume ID [FULL19]? [Y]

Again after a number of diskettes, the CLI prompts for the next diskette.

Remove the diskette, insert the next one, and press NEW LINE.

⤵

Again the CLI labels it and the system starts dumping files.

Please remove the diskette

Su) QPRINT DFILE ⤵

The full backup is done. Print the listing file.

Queued ...
Su) DELETE/V DFILE ⤵

Delete the listing file.

Deleted DFILE
Su) ACL @DPJ10 +,WARE ⤵

Restore the old ACL.

Su)

The full backup is done. You can now remove the diskette from unit DPJ10 and store all diskettes safely. Later on, if needed, all files could be restored to the disk via the commands:

) DIR : ⤵
) OPERATOR ON ⤵
) SUPERUSER ON ⤵
Su) ACL @DPJ10 OP,OWARE ⤵
Su) LOAD/V/R @LFD:FULL01:BACKUP ⤵

Diskette Backup Macros

CLI macros for full and incremental backup of diskettes are shown in Figures 5–5 and 5–6. Diskette backup is supported only under the 16–bit CLI; diskette backup is not possible if you are running the 32–bit CLI. A macro to restore data to diskettes follows later on, in Figure 5–7. Data General shipped these macros with the operating system to ease backup on systems without tape. You must change a specified character as described in the macros (type FULL_BACKUP) before they will work.

The backup macros do not turn Operator mode on, so you will need to turn it on before using a macro. The first time you create any full or incremental diskette fileset, we suggest that you turn Operator on with the /LABEL switch to avoid extra keystrokes. Since you need to label each diskette the first time through, you can save typing effort by using the /LABEL switch.

To reuse a diskette set, you should use Operator mode without the /LABEL switch, so the CLI will check labels and allow you to change diskettes if you make a mistake and insert the wrong diskette. If you need more diskettes (as you probably will), the CLI will let you label and use them as needed.

The DUMP command in the macros specifies a retention period of 0 days, allowing you to reuse the diskette set immediately. You may want to use a nonzero retention period for the macros. For example, you might use /RETAIN=7 for full backup and /RETAIN=2 for incremental backup. (If you ever want to reuse a diskette set before its expiration date, you will need to have the CLI relabel all diskettes.) If desired, you can use a text editor to change the /RETAIN= numbers.

It's best to use Operator mode without the /LABEL switch whenever you can. With OPERATOR and /LABEL, the CLI will label any diskette without asking for confirmation. So it's very important to avoid mixing up diskette sets. Keep the sets separate, and make sure their paper labels are current and accurate:

CAUTION: *Using the /LABEL switch to label a diskette effectively destroys data stored on that diskette and all subsequent diskettes.*

The backup macros use the following volids for diskettes:

FULL_BACKUP	INC_BACKUP
FULL0	INC01
FULL02	INC02
...	...
FULLn	INCn

These are good, descriptive general–purpose volids. They allow you to restore using volid FULL01 (to restore all files) or INC01 (to restore an incremental backup). If you want different volids, use a text editor to change the name FULL01 or INC01 within the macro file. The CLI will then create volids based on yours. To restore, you'll specify your volid, instead of FULL01 or INC01, when you start the RESTORE macro.

Do not change the filename of any of the backup (or restoration) macros.

The incremental backup macro depends on a file named LAST_BACKUP, which is created by the last macro (full or incremental), to run. If an error occurs and the backup is not valid, this file will contain an invalid last backup date. So if ever a backup is invalid (for example, if you abort it and don't restart it that day), you should delete the file LAST_BACKUP. Then copy file LAST_BACKUP.BU to LAST_BACKUP, for example, COPY LAST_BACKUP LAST_BACKUP.BU.

Generally, you should standardize backup procedures. This means reusing volids, keeping each backup diskette set discrete, and keeping a paper label with the filename and volid/sequence number on each diskette. Also on the label of first diskette in a set, you should write the date the diskette was last written to. This date covers the whole set.

Users without Superuser privilege can use these macros to back up and restore their own directories. For nonprivileged users to use these macros, the unit ACL must be something like +,WARE (perhaps set via the UP macro, as described earlier, in the section "Diskette Access Control").

The FULL_BACKUP.CLI Macro for Diskettes

```
[!equal,1,2]
comment This macro does a full labeled diskette backup from the
comment working directory. To do a full system backup, it requires
comment the process that runs it to be PID 2.
push
prompt pop
comment Check for arguments — none is allowed.
[!nequal,%1–%,]
 write
 write This macro backs up copyable files in and below [!DIRECTORY] —
 write excluding DG–supplied files in the root and excluding
 write directories HELP,,PATCH,,and,,UTIL.
 write It doesn't allow arguments. Please try again via ,, %0% ,,
 write when ready.
 write
[!else]
 string ok_to_proceed
 comment Check for the root directory — if so the PID must be 002.
 class(1 2) ignore
 [!equal,[!directory],:]
  [!UEQ,[!pid],002]
   superuser on
  [!else]
   string non_master
  [!end]
 [!else]
  comment Not in the root – check for write access. If we can create
  comment and read from a file, assume we have needed access.
  permanence =?[!pid].[!username].tmp off
  delete =?[!pid].[!username].tmp
  write/l==?[!pid].[!username].tmp test
  string [=?[!pid].[!username].tmp]
  delete =?[!pid].[!username].tmp
  [!equal,[!string],test]
   string ok_to_proceed
  [!else]
   string no_access
  [!end]
 [!end]
 comment If it's not ok to proceed, describe error and stop.
 [!nequal,[!string],ok_to_proceed]
  [!equal,[!string],non_master]
   Write Only the master CLI can back up from the root.
```

Figure 5–5 FULL_BACKUP.CLI Macro for a Full Labeled Diskette Backup (continued)

```
 [!else]
  [!equal,[!string],no_access]
          write
          write Error – [!username] does not have write access to
          write [!directory].
          write ,,,,,,,,You cannot back up this directory.
          write
  [!end]
 [!end]
 pop
[!else]
 comment Check if CLI operator mode is on. If not, stop with message.
 permanence =?[!pid].[!username].tmp off
 delete =?[!pid].[!username].tmp
 operator/l==?[!pid].[!username].tmp
 string [=?[!pid].[!username].tmp]
 delete =?[!pid].[!username].tmp
 [!nequal,([!string]),(OFF)]
  comment Operator mode is on — proceed.
  class1 error
  class2 warning
  write
  write ** Full backup from directory [!DIRECTORY] at
  write [!TIME] on [!DATE] **
  write
  write Please insert the first diskette to receive backup material
  write in the primary — rightmost — unit. The default first–volid
  write for full backups is FULL01. The 16–bit CLI can label
  write diskettes if needed. The first diskette and any others used
  write for backup will be overwritten — so don't use diskettes
  write that have material you want to keep.
  write
  write Please number the paper label of each diskette as it is
  write filled so that — if needed — the diskettes can be restored
  write in correct order.
  write
  write ,,,,,,,,,,,,,,,,,,,,  — Beginning file backup —
  [!equal,[!directory],:]
   comment Backup from root directory. Set diskette unit ACL
   comment for exclusive access, then do the backup and reset ACL.
   acl  @dpj10  op, oware
   DUMP/RETAIN=0/V%0/L%   @LFD:FULL01:BACKUP   &
         #\HELP\NET\PATCH\SYSGEN\UTIL\SWAP\PAGE\PER\PROC\&QUEUE&
 \SWAP.SWAP\?+.BRK\+.ED\?+.JOB\+.LS\?+.TMP       NET:NETGEN:+
```

Figure 5–5 FULL_BACKUP.CLI Macro for a Full Labeled Diskette Backup (continued)

```
    comment  Note: If you want to back up files in HELP, SYSGEN, or
    comment  UTIL, insert the pathname(s) in the macro BEFORE the
          comment  #\HELP... exclusion. For example, the template
          comment  UTIL:MYDATA+ before & in the DUMP command line will
    comment  back up all :UTIL:MYDATA+ files.
    acl @dpj10 +,ware
   [!else]
    DUMP/RETAIN=0/V%0/L%   @LFD:FULL01:BACKUP  #
   [!end]
   write
   write ** Full backup of [!DIRECTORY] complete at [!TIME] **
   permanence/2=ignore  =LAST_BACKUP.BU  OFF
   delete/2=ignore  =LAST_BACKUP.BU
   rename/2=ignore  LAST_BACKUP  LAST_BACKUP.BU
   write/l==LAST_BACKUP   [!DATE]:[!TIME]

   write
   write This backup has created file LAST_BACKUP in this directory
   write for future backups. Don't delete this file.
  [!else]
   write Operator mode is not on. Turn OPERATOR ON and retry the
   write command.
  [!end]
 [!end]
[!end]
[!else]
 write This macro is nonexecutable. You can make it executable by
 write changing the 2 to a 1 in line 1 — make both numbers 1.
 write
 write Backup is very important. Before using the edited macro
 write for backup you should understand how it works. Please test this
 write macro. Use it to back up files. Then try restoring files using
 write the RESTORE macro — before relying on this macro routinely.
[!end]
```

Figure 5–5 FULL_BACKUP.CLI Macro for a Full Labeled Diskette Backup (concluded)

The INC_BACKUP.CLI Macro for Diskettes

```
[!equal,1,2]
comment This macro does an incremental labeled diskette backup from the
comment working directory — based on the last backup, stored in file
comment LAST_BACKUP. To back up from the root directory, the macro
comment requires the process that runs it to be PID 2.
push
prompt pop
comment Check for arguments — none is allowed.
[!nequal,%1-%,]
 write
 write This macro doesn't allow arguments. Please try again by typing
 write ,, %0% ,, when ready.
 write
[!else]
 string ok_to_proceed
 comment Check for the root directory — if so the PID must be 002.
 class(1 2) ignore
 [!equal,[!directory],:]
  [!UEQ,[!pid],002]
   superuser on
  [!else]
   string non_master
  [!end]
 [!else]
  comment Not in the root - check for write access. If we can create
  comment a file, assume write access.
  permanence =?[!pid].[!username].tmp off
  delete =?[!pid].[!username].tmp
  write/l==?[!pid].[!username].tmp test
  string [=?[!pid].[!username].tmp]
  delete =?[!pid].[!username].tmp
  [!equal,[!string],test]
   string ok_to_proceed
  [!else]
   string no_access
  [!end]
 [!end]
 comment If it's not okay to proceed, describe error and stop.
 [!nequal,[!string],ok_to_proceed]
  [!equal,[!string],non_master]
   Write Error - only the master CLI can back up from the root.
  [!else]
   [!equal,[!string],no_access]
```

Figure 5-6 INC_BACKUP.CLI Macro for an Incremental Labeled Diskette Backup (continued)

```
    write
            write Error – [!username] does not have write access to
            write [!directory].
            write ,,,,,,,,You cannot back up this directory.
            write
   [!end]
  [!end]
  pop
 [!else]
  [!equal,[!pathname =LAST_BACKUP],]
   write
   write Error – Incremental backup requires file LAST_BACKUP —
   write ,,,,,,,,, which does not exist. Suggest either doing
   write ,,,,,,,,, FULL_BACKUP or retry incremental backup from
   write ,,,,,,,,, directory in which you have done your most
   write ,,,,,,,,, recent backup.
   write
  [!else]
   comment Check to see if CLI operator mode is on. If not,
   comment stop with message.
   permanence =?[!pid].[!username].tmp off
   delete =?[!pid].[!username].tmp
   operator/l==?[!pid].[!username].tmp
   string [=?[!pid].[!username].tmp]
   delete =?[!pid].[!username].tmp
   [!nequal,([!string]),(OFF)]
    comment Operator mode is on — proceed.
    class1 error
            class2 warning
    write
    write ** Incremental backup from directory [!DIRECTORY]&
                        at [!TIME] on [!DATE] **
    string [=LAST_BACKUP]
    write This backup will dump all files created or modified
            write since [!string].
    write
    write Please insert the first diskette to receive backup
    write material in the primary — rightmost — unit. The default
            write first–volid for full backups is INC01. The 16–bit CLI
            write can label diskettes if needed. The first diskette and any
    write others used for backup will be overwritten — so don't
    write use diskettes that have material you want to keep.
    write
   write Please number the paper label of each diskette as it is
    write filled so that — if needed — the diskettes can be
    write restored in correct order.
```

Figure 5–6 INC_BACKUP.CLI Macro for an Incremental Labeled Diskette Backup (continued)

```
      write
      write ,,,,,,,,,,,,,,, — Beginning file backup —
      [!equal,[!directory],:]
       comment Backup from root directory. Set the diskette unit ACL
       comment for exclusive access, then do backup and reset ACL.
       acl  @dpj10  op, oware
      DUMP/RETAIN=0/AFTER/TLM=[!STRING]/V%0/L%  @LFD:INC01:BACKUP &
            #\NET\PAGE\PER\PROC\QUEUE\SWAP\SWAP.SWAP\SYSGEN
\?+.BRK\+.ED&\?+.JOB\+.LS\?+.TM   P   NET:NETGEN:+
       acl  @dpj10  +,ware
      [!else]
       DUMP/RETAIN=0/AFTER/TLM=[!STRING]/V%0/L%  @LFD:INC01:BACKUP  #
      [!end]
      write
      write ** Incremental backup of [!DIRECTORY] complete at
            write [!TIME] **
      permanence/2=ignore  =LAST_BACKUP.BU  OFF
      delete/2=ignore  =LAST_BACKUP.BU
      rename/2=ignore  LAST_BACKUP  LAST_BACKUP.BU
      write/l==LAST_BACKUP   [!DATE]:[!TIME]

      write
      write This backup has created file LAST_BACKUP in this directory
      write for future backups. Don't delete this file.
     [!else]
      write Operator mode is not on. Turn OPERATOR ON and retry the
            write command.
     [!end]
    [!end]
  [!end]
[!end]
[!else]
  write This macro is nonexecutable. You can make it executable by
  write changing the 2 to a 1 in line 1 — make both numbers 1.
  write
  write Backup is very important. Before using the edited macro for
  write backup you should understand how it works. Please test this
  write macro. Use FULL_BACKUP and this macro to back up some files.
  write Then try restoring files using the RESTORE macro before relying
  write on this macro set routinely.
[!end]
```

Figure 5–6 INC_BACKUP.CLI Macro for an Incremental Labeled Diskette Backup (concluded)

Diskette Backup Examples

For this example, assume that an AOS/VS system is built in October. Information starts accumulating on it immediately. The person acting as system manager decides on a weekly full backup and a daily incremental backup.

The full backup will occur each Friday (with provision for Friday holidays). Incremental backups will be done the following Monday through Thursday. All backups will be done from the root directory (:).

The first full backup in October takes only 14 diskettes; and the related incremental backups take only 2 diskettes each. The November full backups average 20 diskettes and the incremental backups take 2 or 3 diskettes each.

The next sections show the actions and dialog at the system console for the first Friday in December and for the following first incremental backup, on Monday.

All diskette sets have been labeled — with the default names shown in the sample macros (FULL01 or INC01), filename BACKUP.

The Full Backup

At the sample site, Friday afternoon arrives; time for the full backup.

1. Joan, the person who does backups, prepares to bring the multiuser environment down, making sure that no users will lose work when this happens (described in the manual *Installing, Starting, and Stopping AOS/VS*).

2. She returns to the master CLI, PID 2, on the system console:

) WHO ↵
   PID: 13 OP CON0 :CLI.PR — This CLI is PID 13, not PID 2.

) BYE ↵ — Sign off this CLI.
   AOS/VS II CLI TERMINATING ...

   PID: 2 OP OP :CLI.PR — You are now PID 2 — the master CLI.

3. She brings the multiuser environment down with the DOWN.CLI macro.

) DOWN ↵

   ... (EXEC termination messages) ...

4. She makes the root the working directory:

) DIR : ↵

5. She turns Operator mode on:

) OPERATOR ON ↵

Since the diskettes have already been labeled, and the expiration date (default 0 days from the last backup) has been reached, Joan omits the /LABEL switch from the OPERATOR command.

6. She starts the full backup, including the /L switch to specify a listing file of filenames dumped. (This is your decision, but we recommend it — the listing tells which files were backed up.) She ignores the access control issue, since the multiuser environment is down.

) FULL_BACKUP/L=FILES_BACKED_UP ↵

   *** Full backup from directory : at 16:45:05 on 06–AUG–93 ***
   . . .

   Please insert a diskette if not already inserted
   Unit [@DPJ10] Volume ID [FULL01] ? [Y]

7. She inserts the first full backup diskette, FULL01, in the primary (right) unit.

8. She presses the NEW LINE key.

   — Beginning file backup —
   .
   .
   .

   The system writes filenames backed up to the disk file. The diskette fills up. Each diskette takes about 2 minutes to fill.

   Please insert next diskette
   Unit [@DPJ10] Volume ID [FULLn] ? [Y]

9. She replaces the diskette with the next one and presses NEW LINE.

10. She repeats steps 8 and 9 until she's out of previously labeled diskettes. Let's say there are 22 of these, and she has used them all. The system displays

    Please insert next diskette
    Unit [@DPJ10] Volume ID [FULL23] ? [Y]

11. Joan removes the diskette, inserts an unlabeled, hardware formatted diskette, and presses NEW LINE.

    An unlabeled diskette has been inserted
    Do you want to relabel this diskette? [N] Y

12. She does want to relabel this diskette, so she types Y and presses NEW LINE. The system responds

    **** Relabeled diskette *** New Volume ID: FULL23*

13. She repeats steps 11 and 12 as long as the CLI prompts for another diskette. Let's say that only one additional diskette (FULL23) is needed. Some files are copied to the diskette. Then, finally, the system displays:

    Please remove the diskette

    *** Full backup of directory : complete at ... on 06–AUG–93*

)

    The full backup is done. The listing file (shown here as FILES_BACKED_UP) remains in the root directory. You can print it (when the multiuser environment is up) and delete it as desired. Each backup listing should be stored in the same place as its diskette fileset. If you specify a listing file each time you do a full backup, simply delete the old one before starting the next full backup.

14. Joan stores all the full backup diskettes safely in order, in their outer covers, and away from strong magnetic fields.

After a backup, the operating system can be shut down or the multiuser environment can be brought back via the UP.CLI macro.

The Incremental Backup

At the sample site, Monday afternoon rolls around — time for the incremental backup.

1. As she did last week, Joan prepares to bring the multiuser environment down, making sure that no users will lose work.

2. She returns to the master CLI, PID 2, on the system console:

) WHO ↵
    PID: 13 OP CON0 :CLI.PR — This CLI is PID 13, not PID 2.

) BYE ↵ — Joan signs off this CLI.

    AOS/VS II CLI TERMINATING ...

    PID: 2 OP OP :CLI.PR — Now she is PID 2 — the master CLI.

3. She brings the multiuser environment down by executing the DOWN.CLI macro.

) DOWN ↵

    ... (System termination messages) ...

4. She makes the root the working directory:

) DIR : ↵

5. She turns Operator mode on:

) OPERATOR ON ↵

   Since the diskette set has been labeled, and the retention period (0 days in the INC_BACKUP macro shown) has expired, Joan omits the /LABEL switch.

6. She starts the incremental backup, using the /L switch to specify another listing file of filenames copied. We recommend that you use the /L switch when restoring incremental backups.

) INC_BACKUP/L=FILES_BACKED_UP.AUG.09 ↵

   *** Incremental backup from directory : at .... on 09–AUG–93.*

   This backup will dump all files created or modified since 06–AUG–93 ...

   ... (Various system messages) ...

   Please insert a diskette if not already inserted
   Unit [@DPJ10] Volume ID [INC01] ? [Y]

7. Joan gets the first backup diskette, volid INC01, and inserts it in the primary (right) unit.

8. She press the NEW LINE key.

   — Beginning file backup —

   ... (It writes filenames backed up to the disk file) ...

   Please insert next diskette
   Unit [@DPJ10] Volume ID [INCn] ? [Y]

9. Joan removes the diskette, and writes the date on its paper label (using a felt–tipped pen to avoid scoring the diskette surface). Then she replaces the diskette in its envelope and inserts the next diskette in the unit.

10. She repeats steps 8 and 9 as long as the CLI prompts for another diskette. Let's assume she's on the third diskette. Some files are dumped to the diskette. Finally the system responds

    Please remove the diskette

    *** Incremental backup of directory : complete at ... on 09–AUG–93*
)

    This incremental backup is done. Additional diskettes (as shown for the full backup) were not needed. The listing file (shown here as FILES_BACKED_UP.DEC.10) remains in the root directory. It can be printed (when the multiuser environment is up) and deleted as desired. AOS/VS can be shut down or it can stay up.

11. Joan stores all the incremental backup diskettes safely in order, in their outer covers, and away from strong magnetic fields.

The next day, Tuesday, Joan does another incremental backup. The steps are the same as the last incremental backup, except that the date differs and she uses a different listing filename.

As you can see, the procedure is methodical and repetitive, but it can be extremely important.

Restoration Macro for Diskettes

Macro RESTORE.CLI, Figure 5–7, restores material from either a full or incremental backup. The macro allows templates. It requires the backup set filename BACKUP (created by the backup macros, earlier). It also requires that you supply the volume ID of the first diskette in the command line, for example,

```
) RESTORE VOL1 ↵
```

The RESTORE.CLI Macro for Diskettes

```
[!equal,1,2]
comment This macro restores files from either a full or incremental
comment labeled diskette backup. To restore into the root directory,
comment the macro requires the process that runs it to be PID 2.
push
prompt pop
comment Check for at least one argument.
[!equal,%1%,]
 write
 write This macro requires at least one argument: the volid of the
 write first diskette in the fileset. The volid used by the FULL_BACKUP
 write macro is FULL01 and the first volid used by the INC_BACKUP macro
 write is INC01. After the volid argument, you can specify one or more
 write templates. If you omit templates, all files in the fileset will
 write be restored.
 write Run the macro via  ,, %0% ,, volid ,, {template} ...,,
 write when ready.
 write
[!else]
 string ok_to_proceed
 comment Check for the root directory — if so the PID must be 002.
 class(1 2) ignore
 [!equal,[!directory],:]
  [!UEQ,[!pid],002]
   superuser on
  [!else]
   string non_master
  [!end]
 [!else]
  comment Not in the root – check for write access. If we can
  comment create and read from a file, assume we have needed
  comment access.
  permanence =?[!pid].[!username].tmp off
  delete =?[!pid].[!username].tmp
  write/l==?[!pid].[!username].tmp test
  string [=?[!pid].[!username].tmp]
  delete =?[!pid].[!username].tmp
  [!equal,[!string],test]
   string ok_to_proceed
  [!else]
   string no_access
  [!end]
 [!end]
```

Figure 5–7 RESTORE.CLI Macro to Restore Files from Labeled Diskettes (continued)

```
comment If it's not ok to proceed, describe error and stop.
[!nequal,[!string],ok_to_proceed]
 [!equal,[!string],non_master]
  Write Only the master CLI can restore the root directory.
 [!else]
  [!equal,[!string],no_access]
          write
          write Error – [!username] does not have write access to
          write [!directory].
          write ,,,,,,,,,You need Superuser on to restore this
          write directory.
          write
  [!end]
 [!end]
 pop
[!else]
 comment Check if CLI operator mode is on. If not, stop with
 comment message.
 permanence =?[!pid].[!username].tmp off
 delete =?[!pid].[!username].tmp
 operator/l==?[!pid].[!username].tmp
 string [=?[!pid].[!username].tmp]
 delete =?[!pid].[!username].tmp
 [!nequal,([!string]),(OFF)]
  comment Operator mode is on — proceed.
  class1 error
  class2 warning
  write
  write ** Restoration within directory [!DIRECTORY] at [!TIME]
  write on [!DATE] **
  write
  write Please insert the first diskette of the backup fileset
  write in the primary — right — unit.
  write
  write Later, you'll insert diskettes in the same order in which
  write these diskettes were originally backed up.
  write
  write ,,,,,,,,, — Beginning file restoration —
  comment  Restore the fileset, using the supplied volid, filename
  comment  BACKUP, and any supplied template(s).
  [!UEQ,[!pid],002]
   acl @dpj10 op,oware
  [!end]
  LOAD/V/RECENT%0/L%    @LFD:%1%:BACKUP  &
     [!equal,%2%,]#[!else]%2–%[!end]
```

Figure 5–7 RESTORE.CLI Macro to Restore Files from Labeled Diskettes (continued)

```
    write
    write ** Restoration of [!DIRECTORY] complete at [!TIME] **
    write
    [!UEQ,[!pid],002]
     acl @dpj10 +,ware
    [!end]
   [!else]
    write Operator mode is not on.
    write Turn OPERATOR ON and retry the command.
   [!end]
  [!end]
 [!end]
 [!else]
  write This macro is nonexecutable. You can make it executable by
  write changing the 2 to a 1 in line 1 — make both numbers 1.
  write
  write Backup is very important. Before using the edited macro
  write for backup you should understand how it works.
  write Please test this macro. Use FULL_BACKUP and INC_BACKUP
  write to back up some files. Then try restoring files using
  write this macro before relying on this macro set routinely.
 [!end]
```

Figure 5–7 RESTORE.CLI Macro to Restore Files from Labeled Diskettes (concluded)

Restoring Files from Diskettes

Restoring falls into two categories: restoring one or more files, and restoring one or more LDUs. The first category is more common, easier, and faster.

Restoring One or More Files

Usually people restore one or more files when someone has accidentally deleted a file (perhaps a directory), or group of files. Perhaps someone was careless with the DELETE command and a template character. If you want to restore files that were backed up to diskette, there are two things to consider — the tape set(s) needed, and the pathname template(s).

The diskette set you use to restore depends on the date that the lost file(s) was last modified. If the file(s) was created since the last backup, then it wasn't backed up, and cannot be restored. Otherwise, use the backup (incremental or full) that occurred soonest after the file(s) was modified.

If you can't determine when the file(s) was last modified, check the backup listings. If the name of a lost file appears in any listing, then you know the file is in that backup. In the worst case, without a listing or dates, you must restore the last full backup set, then the earliest incremental backup, then next the incremental backup, and so on. This is a good reason to keep your backup listings — especially for incremental backups.

After deciding on the diskette set, you must choose one or more pathname templates (unless you want to restore the entire backup). The RESTORE macro allows template arguments, in which you can specify directory/pathname template(s).

You can restore all files in and below a directory (including subordinate directories) with the template

pathname–from–root:

You can restore all files in and below a user's directory with the template

UDD:username:# (Omit the leading : from UDD.)

For CEO files, the directory structure and restoration procedure differs from that for standard AOS/VS II files. To restore CEO files, see the manual *Managing the CEO® System.*

File Restoration Example

As an example, assume Andy accidentally deleted two files named REPORT.MAY and SUMMARY. He last modified them a week ago. (If lost files were last modified on different dates, you may have to restore them from the last full backup). Acting as operator, you find and get the appropriate diskette set — which happens to be from an incremental backup.

You need a template for the restoration. The two filenames have the letters MA in common. So you could use the template

UDD:ANDY:#:+MA+

This template would work, but it might restore many matching, unwanted files. You can be more specific. The two names each have at least one character, and no period, following the A. So you could refine the template to

UDD:ANDY:#:+MA*–

If you know which directory the files were in, you can use the specific pathnames,

pathname–from–root:REPORT.MAY pathname–from–root:SUMMARY

Let's say you decide on the most general course the +MA+ template — to cover many possibilities.

You go to the system console, get back to PID 2 (perhaps by typing BYE and pressing NEW LINE, if needed). Then you type

) RESTORE INC01 UDD:ANDY:#:+MA+ ↵	Start the macro, specifying an incremental backup and the desired template.
Restoration with in directory : at 18:33 ...	Macro prompts for the next steps.
Please insert the first diskette of the ...	
Please insert a diskette if not already ...	The CLI prompts for a diskette.

Unit [@DPJ10] Volume ID [INC00] ? [Y] ↵	Make sure the correct diskette is inserted and press NEW LINE.
	The CLI runs through the diskette, looking for matching files. Since you specified no listing file, it will display matching pathnames that it restores on the screen.
Please insert next diskette *Unit [@DPJ10] Volume ID [INC01] ? [Y]*	The CLI prompts for the next diskette.
↵	Make sure the correct diskette is inserted and press NEW LINE.

... (System spins though the diskette, looking for files) ...

UDD:ANDY:SUMMARY *UDD:ANDY:BDIR:SUMMARIES* *UDD:ANDY:BDIR:SUMMARIES.ED*	It restores one of the files you want, and some unwanted files.
Please insert the next diskette *Unit [@DPJ10] Volume ID [INC02] ? [Y]*	The CLI prompts for the next diskette.
↵	Insert the diskette and press NEW LINE.

Time passes ...

:UDD:ANDY:REPORTS:REPORT.MAY *:UDD:ANDY:REPORTS:REPORT.MAY.BU* *:UDD:ANDY:REPORTS:REPORT.MAY.ED*	It restores the other lost file, and some other files.
Please insert next diskette *Unit [@DPJ10] Volume ID [INC02] ? [Y]*	The CLI prompts for the next diskette.
CTRL–C CTRL–A	Since all the lost files were restored, you can interrupt the restore by pressing CTRL–C CTRL–A; you're done.

ERROR
CONSOLE INTERRUPT

Restoration of : complete at 18:48:06
)

With the files restored, tell Andy to check them, and to check his directories for unwanted, restored files. He can then delete the unwanted files.

(If Andy had created the files in the full CEO system, not the CEO Word Processor — and had accidentally deleted documents named REPORT.MAY and SUMMARY, you could suggest that he check his CEO wastebasket. In the full CEO system, documents are not actually deleted until a program called Janitor has been run. Until then, documents that people delete can be retrieved from their wastebaskets. If Andy's documents are not in his wastebasket, you can restore them from the last backup as described in the manual *Managing the CEO® System*.)

Shortening a Restoration

When you mention a specific filename (like UDD:SAM:MYFILE) in a restoration, the restoration will end and the CLI prompt will return as soon as the file has been copied to disk.

The restoration will take longer if you use a template like UDD:SAM:MYF+ (as shown in the example) because the system will continue through the last tape, even after the desired file has been copied. The search continues through the last tape because AOS/VS II can't tell, until it reaches the end of the last tape, that there is no matching filename in the tape set. When you see that the files you want have been restored, there's no need to restore others; you can interrupt the restoration as shown above.

It is always desirable to give a specific filename, if you can. If you do, be sure it's the correct one. If you make a mistake with a specific filename, the system will take you all the way through the diskette set and not restore the file you want restored. If you think this may be happening, cancel the restoration with CTRL–C CTRL–A. Then start again.

Restoring an Entire LDU Using Diskettes

The time may come when you need to restore all backup material to an LDU. You may have to restore an entire LDU when a disk wears or fails in such a way that the system can no longer read it.

If you have checked with Data General, and either acquired a new disk or decided to rebuild the old one, follow these steps:

To restore the master LDU, follow the steps listed below. To restore a nonmaster LDU, skip to step 5.

1. Get the tailored system diskette you made after testing your AOS/VS system. (If you don't have a tailored system diskette, use the latest system diskette you received from DG; later, you'll need to generate a tailored system.)

2. Turn to the appropriate chapter in the *Installing* manual for your particular operating system and execute all the numbered steps there, using your tailored system diskette or the DG diskette. (If you don't use a tailored system diskette, generate a tailored AOS/VS system, patch it, start it, and make a system diskette as described in the *Installing* manual for your particular operating system.

3. If you have any DG software products that the macros didn't back up, (those under :UTIL, for example), install these products as described in the software manuals.

4. Use PREDITOR to create an operator profile; then bring up EXEC and create the line printer/batch queues (as described in the *Installing* manual for your particular operating system.

5. To restore a nonmaster LDU, make sure this LDU is formatted with the Disk Formatter utility and has the desired name. Then initialize it into your system as usual.

   If the nonmaster LDU needs DG software installed (for example, FORTRAN 77), and that software was not backed up, make the nonmaster LDU the working directory and install the software from the release media.

6. Get all your incremental backup sets and your last full backup set of diskettes.

7. Log on a user console that's physically close to the system console. Restore your most recently dumped incremental backup diskette set. Proceed backwards through the incremental backups until you have restored them all. Then restore the most recent full backup.

   If you see a *Control point directory max size exceeded* message, the LDU is full. You, (or other users) must delete some previously deleted (restored) files to free some space before you can continue the restoration process.

   The files to delete may be in user directories, where they can be deleted with the CLI DELETE command or in the CEO system, where they can be deleted with option 2, "Delete" on the CEO Documents screen. In the full CEO system, you will also need to run the Janitor to complete the deletions.

   When you have some disk space free, 1000–2000 blocks or more (type SPACE : to find out how much free space), proceed with the next restoration.

8. Make sure you return all diskettes to their covers and store them safely.

9. You're done! You've recreated the entire LDU(s) — with luck, losing only a little work (the files created or changes made since the last backup occurred).

If you have XODIAC networking software, the :NET:NETGEN files were backed up and have been restored; you can regenerate host and RMA files by running NETGEN and specifying the network specification filename.

The RESTORE macro restores files with their original creation times, but it changes the time last modified to the time you restore the files. This means that your next backup must be a full backup, via the FULL_DUMP macro, since the dates last modified aren't real. It also means that the FILESTATUS command with the /BEFORE/TLM= switches won't help identify "old" files after the restoration.

End of Chapter

Chapter 6
Using FSCOPY to Back Up AOS/VS II LDUs and to Restore LDUs or Files

Read this chapter

- When you want to back up AOS/VS II LDUs; or,
- When you want to restore LDUs or individual files.

FSCOPY Features

FSCOPY is an AOS/VS II backup and recovery utility that is part of AOS/VS II Revision 3.01. FSCOPY is optimized to work with large disks, including disk–array storage systems, and with tape–array storage systems. You can use it, however, with any disk or tape. FSCOPY provides the following benefits:

- provides on–line consistent backup of up to 20 Gbytes of data in an operator shift
- users can continue to work while the backup is in progress
- restores LDUs or individual files
- streams tape–array storage systems without monopolizing system resources
- writes ANSI tape labels
- provides backup/restoration status and statistics

FSCOPY backs up initialized AOS/VS II LDUs, which means that users can continue to work while the backup occurs. FSCOPY provides a consistent backup because it backs up files just as they were when the backup began.

You can use FSCOPY as your backup/restore utility of choice. You may want to use FSCOPY for full backups and DUMP_II/LOAD_II for incremental backups.

FSCOPY works with any disk or tape. If you are using it with a tape–array storage system, each volume can be up to seven cartridge tapes. See the manual *The CLARiiON™ Tape–Array Storage System with the DG/UX or AOS/VS II Operating System*.

This chapter explains how to use FSCOPY. For more information about backup in general and the uses of other backup utilities, see Chapter 4.

How FSCOPY Works

FSCOPY provides three major functions: backing up LDUs and restoring LDUs and files. This section gives a simplified overview of how FSCOPY works.

Backing Up an LDU

When you execute it with the /BACKUP switch, FSCOPY reads the bitmap of the LDU you are backing up and then backs up all allocated disk blocks as they were at this date and time (these are called **backed up blocks**). FSCOPY also copies blocks users are about to modify before they are actually modified by the filing system (these are called **copied blocks**). An FSCOPY backup is a "snapshot" of the disk when the backup started: files created or edited after the backup begins are not part of the backup. (If you use the /TIMESTAMP switch, FSCOPY creates a file, FSCOPY_TLB by default, whose date and time of creation show when the backup started. You can use this file when doing incremental backups. See "Using FSCOPY_TLB When Doing Incremental Backups," later in this chapter.)

Restoring an LDU

To restore an LDU, FSCOPY must have exclusive access to the disk. FSCOPY simply writes data from tape to disk, reconstructing the LDU as it existed at the time the backup started. When FSCOPY finishes, you must reinitialize the LDU. Because an LDU cannot be accessed while it is being restored, special consideration must be given when using FSCOPY to back up and restore the root. See "Restoring an LDU," later in this chapter, for more information.

Restoring Files

Restoring files is a four–step process:

1. You run FSCOPY to create an index of what is on the backup tape. FSCOPY creates three index files based on a name you specify, adding pathname extensions .BLKS, .FS_DRV, and .TAPE_DRV. For example, INDEX.BLKS, INDEX.FS_DRV, and INDEX.TAPE_DRV.)
2. You run FSCOPY to read the index and to list the pathnames on the backup tape.
3. Using a text editor such as SED, you edit the listing file, selecting the pathnames you want to restore. FSCOPY can restore from 1 to 240 files each time it runs, so you may create one or more files each of which has a maximum of 240 pathnames.
4. You run FSCOPY to restore the pathnames listed in the file(s) you edited in step 3.

Where to Run FSCOPY

You can issue an FSCOPY command line from the system console or from a user terminal. FSCOPY requires that the process that runs it have the Superuser privilege (FSCOPY turns it on for you). For command line formats and examples of how to use FSCOPY, see the sections that follow: "Backing Up an LDU," "Restoring an LDU," and "Restoring Files."

Backing Up an LDU

Preparing for Backup

To prepare for a good backup, bring the system to a stable state. Broadcast a message to users that they should stop editing or compiling. For server-based applications, pause the servers. If databases are involved, checkpoint databases.

An FSCOPY backup tape set includes backed up blocks, copied blocks, and formatting and header information and typically is up to one and one half times larger than the LDU. (If the system is active, there will be more copied blocks.) Set aside enough tapes for the backup.

Starting the Backup

Mount or insert the tape in its unit. (You can premount tapes if you have multiple units and expect to use multiple tapes.)

Then, start the backup, using the following format:

FSCOPY/BACKUP LDU_PATHNAME @tapeunit *[@tapeunit]*...

FSCOPY requires that you have the Superuser privilege and turns it on for you. Your working directory does not matter. FSCOPY restricts you to one LDU per tape set. Unless you specify more than one tapeunit, in which case FSCOPY continues the backup automatically, FSCOPY will prompt you to mount the next volume if required.

For example, to do a backup of the LDU named UDD1, type

) FSCOPY/BACKUP/DISPLAY/STATISTICS/TIMESTAMP :UDD1 @MTJ0 ↵

In this example, we are backing up the LDU UDD1 (a disk–array storage system) to a tape–array storage system. We use the /DISPLAY switch because we want to monitor runtime status, the /STATISTICS switch to get total statistics when FSCOPY finishes, and the /TIMESTAMP switch to create the file FSCOPY_TLB.

If we had premounted tapes on units MTJ0 and MTJ1, the command line would have been:

) FSCOPY/BACKUP/DISPLAY/STATISTICS/TIMESTAMP :UDD1 @MTJ0 @MTJ1 ↵

There are a number of other backup switches that we could specify, but we accept FSCOPY defaults and do not show these switches in our examples. See Table 6–1 for a list of all backup switches. You can abbreviate switch names so long as they are unique. For example, you can use /DISP for the /DISPLAY switch.

FSCOPY writes an ANSI tape label with a 90–day retention period. You cannot change the retention period, although you can use the LABEL program to scratch the tape if you want to reuse it. If a tape has an ANSI label (written by another backup utility or by the LABEL program), FSCOPY will overwrite a tape if there is no expiration date or if the retention period has expired.

CAUTION: *During a backup, FSCOPY cannot recover from hard disk errors or Fatal buffered tape errors. If you get either of these errors, discard the tape and do not reuse it. Restart the dump.*

Restarting Servers

After FSCOPY reads the bitmap, it displays a message that it has started the backup. Once it has, you can restart servers and broadcast a message to users that they can resume their work.

Tuning FSCOPY Backup

A number of FSCOPY backup switches let you modify how FSCOPY performs. The switches are /DISKREQ, /FSBUFFERS, /RECORDSIZE, /TAPEBUFFERS, /TAPEREQ, and /TASKS. Normally, you will never need to specify these switches, for which we have provided reasonable default values. Should you want to modify these switch values, we suggest that you modify one switch value at a time, doubling or halving the value, and timing the response.

Using FSCOPY_TLB When Doing Incremental Backups

You can use the date and time of creation of FSCOPY_TLB when deciding a time *after* which to perform incremental backups using DUMP_II. When you perform a backup, specify the /TIMESTAMP switch. For example, FSCOPY creates file FSCOPY_TLB with a timestamp of 10–MAY–93 at 18:00:00. (Type F/AS FSCOPY_TLB to view the date and time of creation of this file.) The next day, you do an incremental backup using DUMP_II, using the command line

Su) DUMP_II/AFTER/TLM=10–MAY–93:18:00:00 UDD:# ↵

DUMP_II dumps all files in :UDD and below which were modified after 6:00 p.m. on May 10, 1993.

Table 6–1 FSCOPY Backup Switches

Switch	What It Does
/BACKUP	Backs up an LDU.
/DISKREQ=n[1]	Specifies the number of bytes FSCOPY will request at one time from disk during backup. You can specify a number from 8192 through 1048576 bytes (1 Mbyte). The value you specify must be smaller than the value for /TAPEREQ. The default, if you omit this switch, is 64 Kbytes.
/DISPLAY	Displays runtime status.
/FSBUFFERS=n[2]	Specifies the number of buffers for modified data when backing up an LDU. You can specify a number from 1 through 1024. The default, if you omit this switch, is 5.
/NOBITMAP	Does not scan the bitmap for the LDU and therefore copies all blocks on the LDU. Use this switch to measure the throughput of FSCOPY.
/NPROMPT	Terminates the backup if FSCOPY encounters errors that normally produce an interactive prompt. Use this switch when you cannot ensure that someone will be attending the backup.
/RECORDSIZE=n	Specifies the size of the tape record FSCOPY uses during a backup. You can specify a number from 8192 through 32768 bytes. If your system or tape drive has a maximum buffer size of 16384, you must specify 16K. The default, if you omit this switch, is 32768 bytes.
/SPLIT=n	Divides an FSCOPY backup into multiple n–Megabyte physical files on the backup tape. This speeds up file restoration and also reduces data loss if a part of a backup tape becomes unreadable. You can specify a number from 1 through 2097152 Megabytes. The default, if you omit this switch, is 1024 Mbytes (1 Gbyte).
/STATISTICS	Displays statistics when FSCOPY completes.
/TAPEBUFFERS=n[1]	Specifies the number of buffers for copied data when backing up an LDU. You can specify a number from 1 through 1024. The default, if you omit this switch, is 10.
/TAPEREQ=n	Specifies the number of bytes FSCOPY will write to tape at one time. You can specify a number from 8192 through 1048576 bytes (1 Mbyte). The value you specify must be larger than the value for /DISKREQ and must be a multiple of the value for /RECORDSIZE. The default, if you omit this switch, is 224 Kbytes.

(continued)

Table 6–1 FSCOPY Backup Switches

Switch	What It Does
/TASKS=n	Specifies the number of system tasks FSCOPY uses to access an LDU during a backup. You can specify a number from 1 through 60. The default, if you omit this switch, is 10.
/TIMEOUT=n	Specifies the number of seconds FSCOPY waits for an operator response before timing out (aborting). You can specify a number from 1 through 15000 (4 hours and ten minutes). The default, if you omit this switch, is 900 (15 minutes).
/TIMESTAMP*[=name]*	Specifies the name of the file FSCOPY creates whose date and time of creation indicate when the backup occurred. You can specify a regular AOS/VS filename, which FSCOPY creates in the top–level directory of the LDU you are backing up. The default, if you omit name, is FSCOPY_TLB.

1 For all switches that take a value, n, you can specify K for Kbytes or M for Mbytes.

2 (the number of FSBUFFERS + the number of TAPEBUFFERS) x TAPEREQ size (224Kbytes) + .5 Mbyte for FSCOPY itself = total memory required to run FSCOPY, approximately 4 Mbytes.

(concluded)

Monitoring Status

Use the /DISPLAY switch to monitor runtime status. FSCOPY computes the percentage of the backup completed and estimates the amount of time remaining until the backup is done. FSCOPY also displays the tape transfer rate, in bytes/second (B/s), Kbytes/second (KB/s), or Mbytes/s (MB/s). The transfer rate varies, depending on the number of soft errors, the characteristics of the tape drive, and how far along in the backup you are. The accuracy improves as the backup continues. Figure 6–1 gives an example of this screen. FSCOPY displays a similar screen for other activities such as restoring an LDU, creating an index, or creating a list of files.

```
FSCOPY Revision nn.nn                          10-May-93 11:38

                          Backup of UDD1

+================-------------------------------------------------+
|################                                                 |
+================-------------------------------------------------+

Backup 25% complete.  Estimated time remaining: 120 minutes.
                                    Disk        Tape
Transfer Rate:                    200.5KB/s    210.8KB/s
```

Figure 6–1 FSCOPY Backup Status Screen

Getting Backup Statistics

Use the /STATISTICS switch to get statistics after FSCOPY runs. Figure 6–2 gives an example of the screen you can use to examine backup status.

```
Statistics for full volume backup of UDD1
The backup took 1347 seconds to complete.
It required 1 backup tape(s).
                          DISK                    TAPE
Transfer Rate:          747.1KB/s               871.9KB/s
Requests:                5242                    5243

Copied blocks:           0
Backed up blocks:        2012651
```

Figure 6–2 FSCOPY Backup Statistics Screen

FSCOPY reports the time it took to complete the backup, the number of tapes in the backup set, and the transfer rate for disk and tape in bytes/second (B/s), Kbytes/second (KB/s), or Mbytes/s (MB/s), as well as the number of requests for disk and tape and the number of copied blocks and backed up blocks. Here, there was no user activity, so there were no copied blocks.

Restoring an LDU

Preparing for Restoration

Typically, you will need to restore an LDU when a disk has a hard failure (crashes) and you need to replace it. FSCOPY requires that the LDU to be restored be exactly the same size as the LDU it is replacing. FSCOPY will tell you what to do if the sizes are different. If they are, run Disk Jockey and make them the same.

There is one special case: restoring the root (:) directory itself. When restoring the root directory, your working directory must be a different LDU and you must have copied a tailored system to that LDU. For these reasons, it is easier and preferable to work from a tailored system tape to restore the root directory. See the manual *Installing, Starting, and Stopping AOS/VS II* for more information about making a tailored system tape set.

Starting the Restoration

To restore an LDU, FSCOPY must have exclusive ownership of the LDU. If other users have access to the LDU, FSCOPY will warn you that it cannot run. In this case, you can release and reinitialize the LDU, and then run FSCOPY again so that it can gain exclusive ownership. To release an LDU, use the CLI command RELEASE. You may also need to use the /FORCE switch to do this.

Mount or insert the tape in its unit (you can premount tapes if you have multiple units and will be restoring from multiple tapes).

Then start the restoration, using the following format:

FSCOPY/RESTORE LDU_NAME_ON_TAPE LDU_PATHNAME_ON_DISK @tapeunit *[@tapeunit]...*

For example, to restore the LDU named UDD1, type

) SUPERUSER ON ↵

Su) DIR : ↵

Su) RELEASE/FORCERELEASE UDD1 ↵

Su) INITIALIZE/DIR=: @DPJ1 ↵

UDD1

Su) FSCOPY/RESTORE/DISPLAY/STATISTICS UDD1 :UDD1 @MTJ0 ↵

We are restoring the LDU named UDD1 from a high–capacity cartridge tape. We specify UDD1, the *name* of the LDU on the backup tape, and :UDD1, the *pathname* of the directory on disk. We use the /DISPLAY switch because we want to monitor runtime status, and the /STATISTICS switch to see total statistics. There are a number of other switches that you can specify, but we accept FSCOPY defaults and do not show these switches in our examples. See Table 6–2 for a list of all restoration switches.

CAUTION: *During a restoration of an entire LDU, FSCOPY cannot recover from hard disk errors.*

Reinitializing the LDU After FSCOPY Finishes

When FSCOPY finishes restoring the LDU (the CLI prompt returns), it releases the disk. You must reinitialize the LDU into the system.

Table 6–2 FSCOPY Restoration Switches

Switch	What It Does
/DISPLAY	Displays runtime status.
/FILES=filelist	Restores files listed in filelist (when used with /INDEX and /RESTORE).
/INDEX=name	Specifies the name FSCOPY should use when it creates the index files. FSCOPY adds the extensions .BLKS, .FS_DRV, and .TAPE_DRV to name.
/LIST=name	Specifies the name of a disk file to use for listing pathnames on the backup tape (when used with /INDEX). The default, if you omit name, is @LIST.
/NPROMPT	Terminates the restoration if FSCOPY encounters errors that normally produce an interactive prompt. Use this switch when you cannot ensure that someone will be attending the restoration.
/RESTORE	Restores an LDU. Also builds an index of backed up files (when used with /INDEX); lists backed up files (when used with /INDEX and /LIST); and restores files (when used with /INDEX and /FILES).
/STATISTICS	Displays statistics when FSCOPY completes.
/TASKS=n	Specifies the number of tasks FSCOPY uses to access an LDU when restoring an entire LDU. You can specify a number from 1 through 60. The default, if you omit this switch, is 10.
/TIMEOUT=n	Specifies the number of seconds FSCOPY waits for an operator response before timing out (aborting). The default, if you omit this switch, is 900 (15 minutes). You can specify a number from 1 through 15000 (4 hours and ten minutes).

Restoring Files

Restoring files is a four–step process;

1. You run FSCOPY to create an index of what is on the backup tape.
2. You run FSCOPY to generate a list of pathnames on the backup tape.
3. You use a text editor to select the pathnames of files you want to restore.
4. You run FSCOPY to restore these pathnames.

The following sections go into more detail.

Creating an Index

To create an index of what is on the backup tape, perform the following steps.

1. Mount or insert the tape in its unit (you can premount tapes if you have multiple units and will be restoring from multiple tapes).
2. Choose a working directory to simplify issuing subsequent command lines. Here, we make :BACKUP our working directory.

) DIR :BACKUP ↵

3. Run FSCOPY to create an index, using the following format:

   FSCOPY/RESTORE/INDEX=name LDU_NAME @tapeunit *[@tapeunit]* ...

   For example,

) FSCOPY/RESTORE/DISPLAY/INDEX=UDD1_INDEX UDD1 @MTJ0 ↵

   FSCOPY reads the tape set inserted in @MTJ0 and creates the index files UDD1_INDEX.BLKS, UDD1_INDEX.FS_DRV, and UDD1_INDEX.TAPE_DRV in your working directory (here :BACKUP). (The index files use about 4–5% of the size of the LDU.) If these files exist, FSCOPY warns you that you must delete them. If this happens, delete the files and repeat step 3.

Generating a List of Pathnames

To generate a list of the pathnames of the files on the backup tape, run FSCOPY to read the index files, using the following format:

FSCOPY/RESTORE/INDEX=name/LIST=name

For example,

) FSCOPY/RESTORE/DISPLAY/INDEX=UDD1_INDEX/LIST=UDD1_FILES ↵

FSCOPY reads the three index files specified by UDD1_INDEX and creates the file list UDD1_FILES.

Editing the List of Files

The file list FSCOPY creates includes the pathnames of *all* of the files on the backup tape. To restore individual files from the file list, perform the following steps.

1. Use a text editor such as SED to edit the file list. Create one or more files containing the pathnames of files to restore. One way to do this is by copying pathnames to one or more files using the SED command DUPLICATE. For example, to restore Kevin's files, you could create UDD1_KEVIN.

   Each file you create has a maximum of 240 pathnames, with one relative pathname per line. (The pathnames are relative to the top–level directory of the LDU. A relative pathname for FILEA in the directory :UDD1:KEVIN:WORK will appear as =UDD1:KEVIN:WORK:FILEA.)

2. It is best to restore files when users are not active. If this is not possible, advise users that certain directories are "off limits." One way to ensure this is to make the ACL of each parent directory a null. For example, if you want to reload files into :UDD1:KEVIN, type

   Su) ACL :UDD1:KEVIN,, ↵

Restoring Files

Finally, to restore files, follow these steps:

1. Change your working directory to the top–level directory of the LDU where you want to restore files. Then run FSCOPY to restore pathnames listed in the file(s) you created, using the following format:

   FSCOPY/RESTORE/INDEX=name/FILES=name LDU_NAME @tapeunit *[@tapeunit]* ...

   For example,

) DIR :UDD1 ↵

) FSCOPY/RESTORE/DISPLAY/INDEX=UDD1_INDEX/FILES=UDD1_KEVIN UDD1 @MTJ0 ↵

   FSCOPY restores on to the LDU UDD1 the pathnames listed in the disk file UDD1_KEVIN, using the information in the index files specified by UDD1_INDEX.

2. If you changed directory ACLs, make sure that you change them back to what they were. For example,

   Su) ACL :UDD1:KEVIN KEVIN,OWARE +,RE ↵

Restoration Tips

You can save some time by following these suggestions:

1. Create the index files and the file list right after the backup, *before* you actually have to restore files.
2. If you know the pathnames you want to restore, you can use a text editor to create a file list of pathnames (from the root) of the files. Make sure that each pathname begins with an = sign. You can avoid generating the list of files if you do this.
3. You can use /LIST and /FILES in the same command line, but FSCOPY will only restore the first 240 pathnames that it finds.

Monitoring Restoration Status

During a file restoration, the /DISPLAY switch monitors the status of the restoration. Figure 6–3 gives an example of this screen. FSCOPY first shows the pathname of each file it is resolving and informs you as it restores blocks to a particular pathname.

```
FSCOPY Revision nn.nn                          10-May-93  11:38

                    File Restore of UDD1

+=========================-------------------------------------+
|#########################                                     |
+=========================-------------------------------------+

File Restore 44% complete.  Estimated time remaining: 2 minutes

Transfer rate:   100KB/s

Restoring blocks to pathname:  =KEVIN:WORK:FILEA
```

Figure 6–3 FSCOPY Restore Status Screen

Getting Restoration Statistics

The /STATISTICS switch displays statistics after FSCOPY runs. Figure 6–4 gives an example of the screen you can use to examine the status of the restoration.

```
Statistics for file restoration of UDD1
The restoration took 162 seconds to complete.
Transfer rate:          17.4KB/s
Tape requests:          44
Files restored:         31
Blocks restored:        1053
```

Figure 6–4 FSCOPY Restoration Statistics Screen

End of Chapter

Chapter 7
Using LDCOPY to Back Up and Restore AOS/VS II LDUs

Read this chapter

- When you want to back up and restore an AOS/VS II LDU; or,
- When you want to copy an AOS/VS II LDU; or,
- When you want to run LDCOPY using script files.

LDCOPY is an AOS/VS II utility that you can use to back up and recover complete LDUs. You access the LDCOPY utility from the Disk Jockey disk management program's Main Menu. You can also use script files with the standamong version of the Disk Jockey utility.

Using LDCOPY, you can copy:

- disk–to–tape (backup)
- tape–to–disk (restore)
- disk–to–disk (copy)

LDCOPY does not allow tape–to–tape copies.

LDCOPY backs up non–initialized LDUs, which means that users cannot continue to work while the backup occurs. Therefore, an alternative to LDCOPY is FSCOPY, which is described in the previous chapter.

This chapter explains how to use LDCOPY. It assumes that you know how to use the Disk Jockey utility. For more information about the Disk Jockey utility, see the manual *Installing, Starting, and Stopping AOS/VS II.* For more information about backup in general and the uses of other backup utilities, see Chapter 4.

Running LDCOPY

LDCOPY is an LDU copy function that is part of the Disk Jockey utility. As such, you run LDCOPY by first invoking Disk Jockey.

Running LDCOPY from Stand–Alone Disk Jockey

If AOS/VS II is not yet running, selecting option 7, "Enter the Disk Jockey Main Menu," on the Technical Maintenance Menu brings up the stand–alone Disk Jockey, and positions you at the Disk Jockey Main Menu.

The Disk Jockey Main Menu offers you all the options necessary to perform disk management tasks, such as copy an LDU.

Running LDCOPY from Stand–Among Disk Jockey

The stand–among Disk Jockey is in directory :UTIL; its filename is DJ.PR. DJ.PR is among the system files loaded onto the disk when you installed system files.

You can use the stand–among version of Disk Jockey to copy any LDU other than the master LDU or any LDU that is currently initialized. Stand–among Disk Jockey also lets you run it using script files. You can run the stand–among version of Disk Jockey from the system console or any user terminal enabled by EXEC.

In order to use the stand–among version of Disk Jockey, make certain that you are located in the root (:) directory. Set your search list to include :UTIL and :HELP (for on–line Help).

```
) SEA :UTIL,:HELP ⤵
```

Then execute Disk Jockey by typing

```
) XEQ DJ ⤵
```

AOS/VS II will bring up the Disk Jockey Main Menu.

Figure 7–1 shows the Disk Jockey Main Menu.

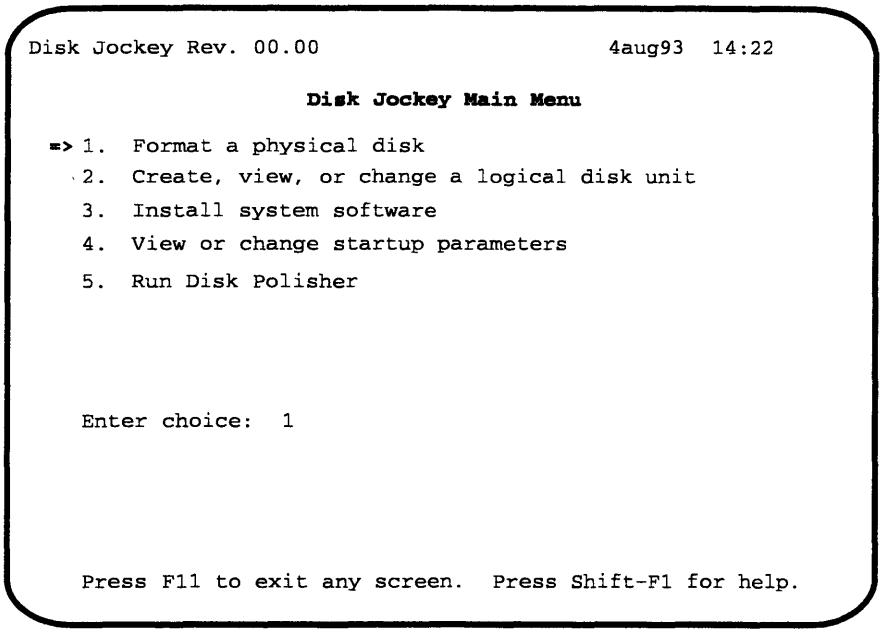


Figure 7–1 Disk Jockey Main Menu

In order to copy an LDU, select option 2, "Create, view, or change a logical disk unit." This displays the LDU Menu. Figure 7–2 shows the LDU Main Menu.

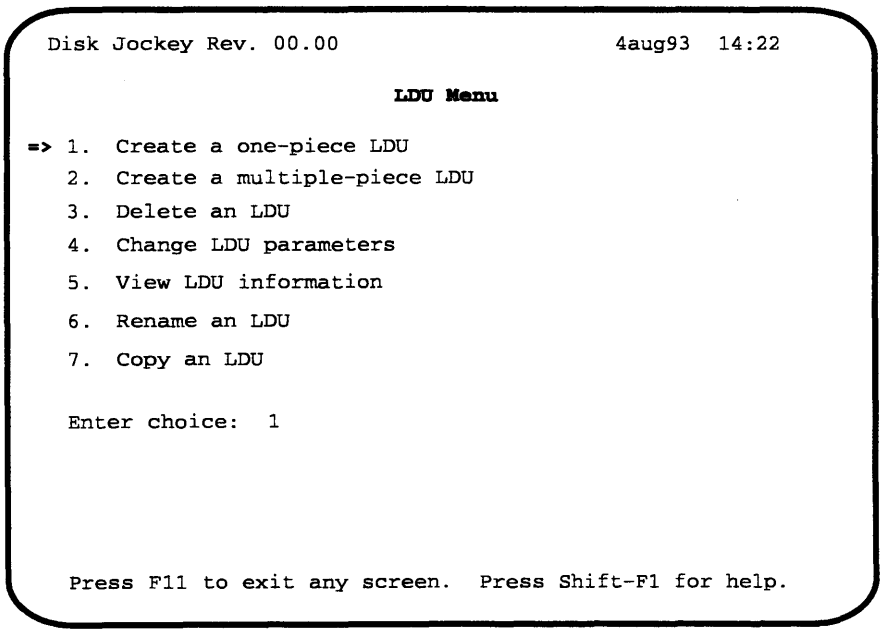


Figure 7–2 LDU Main Menu

You access the LDCOPY utility by selecting option 7, "Copy an LDU," from the LDU Menu. You can also use the keyword, LDCOPY.

The following sections describe how to use option 7, "Copy an LDU" — the LDCOPY utility — to back up and recover your AOS/VS II LDUs.

Copying an LDU (LDCOPY)

An LDU copy can be any combination of the following: disk–to–tape (backup), tape–to–disk (restore), or disk–to disk (copy). AOS/VS II does not allow tape–to–tape copies.

Figure 7–3, below, is the Copy a Logical Disk Unit command screen that Disk Jockey displays when you choose option 7, "Copy an LDU," from the LDU Menu.

```
Disk Jockey Rev. 00.00                          4aug93  14:22

                      Copy A Logical Disk Unit

 Copy from tape or disk? (T = Tape, D = Disk): D
 Unit name(s) to copy from :

 LDU filename  :
 LDU unique ID :

 Verify (only) source tape (Y or N)          :

 *** WARNING: Destination media will be overwritten ***
 Copy to tape or disk? (T = Tape, D = Disk)  :
 Unit name(s) to copy to   :

 LDU Filename  :
 LDU Unique ID :

 Verify tapes immediately (Y or N)  :
 Override tape defaults (Y or N)    :

 Execute? (Y or N):

 Press F11 to exit any screen.  Press Shift-F1 for help.
```

Figure 7–3 LDCOPY Command Screen

The following sections discuss the input fields of the Copy a Logical Disk Unit command screen.

Copy from tape or disk? (T = Tape, D = Disk)

Disk Jockey needs to know whether to read from a tape or disk unit. Type T for magnetic tape, or D for disk, and press NEW LINE.

Unit name(s) to copy from:

Disk Jockey is asking you to specify the disk or tape unit name(s) that contains the LDU to be copied from. To specify mirrored disk units (which are discussed in detail in Chapter 10), separate each unit with an exclamation point (for example, DPJ1!DPJ11); to specify multiple tape units, separate each unit with a semicolon (for example, MTB0;MTB1;MTB2). Type the tape or disk unit name(s), and press NEW LINE. Disk Jockey will display the Device Specification Screen, which lets you confirm your choice of unit name. Figure 7–4 shows this screen.

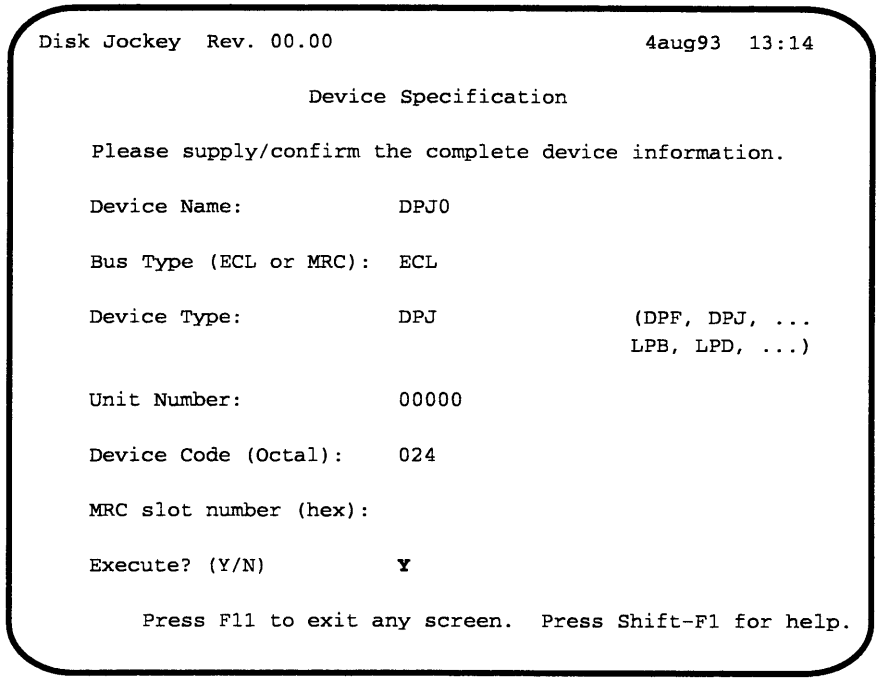


Figure 7–4 Disk Jockey Device Specification Screen

Confirm or change the device information. When you have, Disk Jockey returns to the LDCOPY Command Screen.

LDU filename:

The LDU filename is the name you gave the LDU when you created it, and the name by which you reference this particular LDU. LDU filenames can be as many as 31 characters. Type the filename of the LDU that you want copied, and press NEW LINE. When a tape unit is specified as the from/to source, Disk Jockey skips this question.

LDU unique ID:

Disk Jockey is asking for the unique ID that you specified when you created the LDU image. For mirrored LDUs, specify each image separated by an exclamation point (for example, image1!image2). The image's unique ID can be as many as 31 characters. When a tape unit is specified as the from/to source, Disk Jockey skips this question. Type the image's unique ID, and press NEW LINE.

Verify (only) source tape (Y or N):

This question verifies the source tape without actually doing the copy (similar to executing a LOAD_II/N command from the CLI). If the source is a disk, then this question is automatically skipped. Type N (no) and press NEW LINE if you are copying from tape to disk. Type Y (yes) and press NEW LINE if you want the source tape verified.

Copy to tape or disk? (T = Tape, D = Disk)

Disk Jockey now wants you to specify the tape or disk to which you want the LDU you have specified copied. Type T (tape) if the LDU is to be copied to magnetic tape, or D if the LDU is to be copied to disk, and press NEW LINE.

Unit name(s) to copy to:

At the *Unit name(s) to copy to* prompt, specify the disk or tape unit names(s) (DPJn, MTBn, etc.) to which you want the LDU copied. Appendix A in the manual *Installing, Starting, and Stopping AOS/VS II* lists the standard tape unit names and their device codes. Tables in Chapter 2 of that manual list the standard AOS/VS II disk unit names and their device codes. Type the tape or disk unit name(s), and press NEW LINE.

LDU filename:

The LDU filename is the name you gave the LDU when you created it, and the name by which you refer to this particular LDU. LDU filenames can be as many as 31 characters. Type the filename of the LDU that you want copied, and press NEW LINE. When a tape unit is specified as the from/to source, the LDU filename and unique ID questions are automatically skipped.

LDU unique ID:

Disk Jockey is asking for the unique ID that you specified when you created the LDU image. For mirrored LDUs, specify each image separated by an exclamation point (for example, image1!image2). The image's unique ID can be as many as 31 characters. Type the image's unique ID, and press NEW LINE.

The next two questions apply only if you are using tapes in the copy. If you specify disk units as the source and destination devices, Disk Jockey will position the screen cursor at the *Execute prompt* of the Copy a Logical Disk Unit command screen after you answer the *LDU unique ID* question.

Verify tapes immediately? (Y or N):

If you answer Y (yes) to the *Verify tapes immediately?* question, Disk Jockey will read the tape after the copy to make sure the tape is a good copy. Disk Jockey will report any errors with the tape that it finds. If you answer N (no) to this question, Disk Jockey will not verify the tape after the copy. Disk Jockey will, however, report any hard tape errors it discovers. Depending on whether or not you want Disk Jockey to verify the tape, type Y or N and press NEW LINE. Verification may greatly increase copy time, depending on the size of the LDU being copied. This question is only asked during a disk–to–tape copy.

If you answer No to the *Override tape defaults* question, skip this section, and turn to the explanation of the *Execute?* prompt on the last page of this chapter.

Override tape defaults? (Y or N):

If you answer Y (yes) to the *Override tape defaults?* question, Disk Jockey will display the Change Default Settings for Tape Units command screen shown in Figure 7–5, below. This question will be asked for tape–to–disk and disk–to–tape copies.

```
 Disk Jockey Rev 00.00                    4aug92     12:52
            Change Default Settings for Tape Units

=> Expiration Date [MM/DD/YY]          :

   Premount volume ID's? (Y or N)      :

   Volids :

   Ignore existing labels? (Y or N)    :

   Generation number                   :

   Tape Density                        :

   Buffer size:

   Execute? (Y or N):



   Press F11 to exit any screen.  Press Shift-F1 for help.
```

Figure 7–5 Change Default Settings Command Screen

If you answered No to the *Override tape defaults* question on the Copy a Logical Disk Unit command screen, Disk Jockey will not display the Change Default Setting for Tape Units command screen. Disk Jockey will instead move the menu cursor to the *Execute?* prompt on the Copy a Logical Disk command screen.

If you choose to override existing tape default values by answering Y (yes) to the *Override tape defaults* question on the Copy a Logical Disk Unit command screen, Disk Jockey will display the Change Default Settings for Tape Units command screen shown above, and ask you the following series of questions.

Expiration date [MM/DD/YY]

Disk Jockey is asking for the expiration date of the retention period for labeled tape. AOS/VS II assigns a default retention period of 90 days when you write to a labeled tape. The AOS/VS II DUMP_II utility has a /RETAIN– switch that you can use to change this default value. AOS/VS II will not overwrite a file on a tape until that tape's retention period has expired.

Type the expiration date of the retention period using the form MM/DD/YY (for example, 07/23/92), and press NEW LINE.

Premount volume IDs? (Y or N)

Disk Jockey allows you to specify volids before the copy takes place. You must do this if you know that the LDU copy will use several tape volumes, and you have several tape units that you can use for the copy operation. If you want to premount tape volumes before the copy operation begins, type Y (yes) and press NEW LINE; if you do not want to premount tapes, type N (no) and press NEW LINE.

Volid(s)

Disk Jockey is asking for the volid(s) of each tape volume you want to use. You may enter as many as 20 six–character volids. LDCOPY will use them in the order in which you enter them in this input field. If additional volids are needed, LDCOPY will prompt you for a volid as the additional tape(s) are mounted. You may choose to use volids of more or less than six characters, but LDCOPY will pad or truncate all volids to six characters. All volids must be unique. Separate multiple volids with a semicolon (for example, VOL1;VOL2;VOL3). Type in the six–character alphanumeric volids and press NEW LINE. If you don't specify a volid, LDCOPY assigns the default volid LDC001 to the first tape volume, LDC002 to the second, and so forth. Disk Jockey skips this question if you answered N (no) to the *Premount volume IDs?* question.

Ignore existing labels? (Y or N)

If you answer Y (yes) to the *Ignore existing labels?* question, LDCOPY will ignore the existing tape label information, including the retention period. Type Y (yes) or N (no), and press NEW LINE.

Generation number

The Generation number that Disk Jockey is requesting is used to differentiate between multiple sets of tapes used for the same purpose. For example, you may choose to make multiple backup copies of an LDU. If so, you would need to assign each set of tapes a unique generation number to differentiate among them. The default value for this input field is 1. Either accept the default value of 1 by pressing NEW LINE, or type the correct generation number for the tape that you have mounted, and then press NEW LINE.

Tape density

Disk Jockey is asking what density you want to assign the tape. You can control the density of the tape dumped to when you use a variable–density tape unit, like an MTB. The following are the mode/density options available to you:

Mode	Density
0	Automatic Density Matching (ADM)
800	800 b/in (bits per inch)
1600	1600 b/in
6250	6250 b/in
LOW	Lowest density of the drive
MEDIUM	Middle density of a tri–density drive; lower density of a dual–density drive
HIGH	Highest density of the drive

If you select ADM, AOS/VS II will automatically match the density that the tape was dumped at. ADM is useful when you do not know what density a tape is. If you intend to use a tape on another unit, be careful when selecting LOW, MEDIUM, or HIGH: what is "low," "medium," or "high" on one unit may be different on another unit, which would prevent loading from the tape. Type the correct tape density, LOW, MEDIUM, HIGH, or ADM, and press NEW LINE.

Buffer size

Disk Jockey is asking you to specify the buffer size to be used when dumping to tape; for example, 8192, 16384, or 32768. Choose a number up to the limit set at system generation. ECLIPSE MV/1000, MV/1400, MV/2000, MV/2500 and Data General DS/7500 systems allow a maximum of 16384 bytes. ECLIPSE MV/3000 series systems and above allow a maximum of 32768 bytes. On all systems, 21–, 120–, 150–, 320–, and 525–Mbyte cartridge tapes allow a maximum of 16384 bytes.

Type the buffer size you want the tape to have, and press NEW LINE.

Execute?

When you have specified answers for all the input fields in the Change Default Settings for Tape Units command screen, Disk Jockey will move the screen cursor to the *Execute?* prompt. Type Y (yes) and press NEW LINE. Disk Jockey will then return you to the *Execute?* prompt on the Copy a Logical Disk Unit command screen.

When you have specified answers for all the input fields in the Copy A Logical Disk Unit command screen (including the Change Default Settings for Tape Units command screen if necessary), Disk Jockey will move the screen cursor to the *Execute?* prompt.

If everything on the screen is to your satisfaction, type Y (yes) and press NEW LINE. If you want to change any of the input fields, type N (no) and press NEW LINE; Disk Jockey will move the screen cursor to the first input field of the command screen. To change any of the information, move the screen cursor to the desired input field using the downarrow or uparrow keys, or the NEW LINE key. You can type over any answer you want to change. When you are satisfied with all of the answers on the command screen, move the screen cursor to the *Execute?* prompt, type Y, and press NEW LINE or the EXECUTE function key (F1).

Using LDCOPY Scripts

Standamong LDCOPY can take its input from a script file that you have created instead of requiring you to fill in screen fields. This lets you simplify and possibly to automate your LDCOPY backups by running LDCOPY from a macro unattended and/or off hours.

As with standamong Disk Jockey, you cannot copy the root or an initialized disk using a script file.

The following sections describe

- the LDCOPY switches
- the LDCOPY script file format
- communicating with LDCOPY
- troubleshooting an LDCOPY script file

LDCOPY Switches

When running LDCOPY, use the following format:

DJ/SCRIPT=filename[*/optional switches*]

/SCRIPT=filename	Specifies the script file to use. The next section defines the format for a script file.
/CHECK	Validates all values in the script file, including conflicting identifiers (switches). Use this switch in conjunction with the /SCRIPT switch to validate the script file without performing the LDCOPY operation. LDCOPY will display any errors on your screen and write them to the log file, described later.
/L=pathname	Specifies the pathname for the log file, which can be up to 27 characters long (if you specify more characters, LDCOPY truncates the file name). If the file does not exist, Disk Jockey creates it, If it does exist, disk Jockey appends to it. The log file is a text file which contains the results of the operations specified in the script file as well as status messages and error messages. It is useful for troubleshooting. By default, Disk Jockey creates a log file called <script filename>.LOG in your working directory. If Disk Jockey is unable to create this file, Disk Jockey will terminate with an error. See the section "Understanding the Log File."
/MESSAGE=pid	Sets the destination for LDCOPY messages. By default, Disk Jockey displays prompts and error messages at the terminal where the user started the process when running interactively and to the operator's console (PID 2) when running without a console.
/WAIT	Causes LDCOPY to wait for operator response when it encounters an error. By default, LDCOPY skips over the current operation and continues when it encounters an error.

LDCOPY Script File Format

When creating an LDCOPY script file, observe these rules (see the examples that follow):

1. Exclusive of any comments, a script file must start with a line consisting of LDCOPY in brackets, like this

   [LDCOPY]

   This keyword tells Disk Jockey that subsequent lines contain information for an LDCOPY operation. If you omit this line, Disk Jockey will terminate with an error.

2. You can include more than one [LDCOPY] section in a single script file. Disk Jockey processes each section sequentially.

3. Each line after the [LDCOPY] line has the form identifier=value. Each identifier is derived and slightly modified from the actual questions in LDCOPY screen(s), and most have the same default values. For more information, see the descriptions earlier in this chapter. The identifiers and their default values are:

   From_Unit_Name – no default value.

   From_LDU_Name – no default value. (Value is required if source is a disk.)

   From_LDU_ID – no default value. (Value is required if source is a disk.)

   Verify_Source_Tape – default value is NO.

   To_Unit_Name – no default value.

   To_LDU_Name – no default value. (Value is required if destination is a disk.)

   To_LDU_ID – no default value. (Value is required if destination is a disk.)

   Verify_Tape – default value is NO.

   Expiration_Date – default value is 90 days from today's date.

   Premount_Volume_ID_List – default value is NO.

   Ignore_Existing_Labels – default value is NO.

   Generation_Number – default value is 0001.

   Tape_Density – default value is ADM.

   Tape_Buffer_Size – default value is 32768.

4. Identifier names and values are case insensitive, and the order of the lines inside each [LDCOPY] section is irrelevant. Enter values just as you would when completing LDCOPY screens.

5. If an identifier is assigned to null (that is, identifier=) or missing, Disk Jockey will use the default value implicitly. If there is no corresponding default value for the identifier, Disk Jockey will return a descriptive error.

6. You can separate the identifier name, the equal sign, and the value with spaces.

7. You can include comment lines and blank lines anywhere in a script file. Start each comment line with a semicolon in column 1.

Figures 7–6 and 7–7 give examples of valid script files.

```
;
; This is a sample script file for LDCOPY
;

[LDCOPY]
From_Unit_Name = dpj1
From_LDU_Name = udd
From_LDU_ID = udd

;
; Copy LDU udd to a mirrored disk - each image is on a single unit
;

To_Unit_Name = dpj1!dpj2
To_LDU_Name = mirrored_ldu
To_LDU_ID = image1!image2
```

Figure 7–6 LDCOPY Script File

```
;
; This sample file has two LDCOPY sections
;

[LDCOPY]

From_UNIT_NAME = MTJ0
Tape_Density = 6250
;
; Copy from tape to a disk
;

To_Unit_Name = dpj1
TO_LDU_Name=test
TO_LDU_ID = test

[LDCOPY]

; Copy from disk to tape

To_Unit_Name = MTB2
Verify_Tape = Yes

From_Unit_Name = DPJ3
From_LDU_Name = Payroll
From_Ldu_ID = OS
```

Figure 7–7 LDCOPY Script File with Two Sessions

Communicating with LDCOPY

When running Disk Jockey with a script file, use the /WAIT switch if you want to interact with LDCOPY when it encounters an error condition. Disk Jockey creates an IPC file called DJ<process number> in your initial working directory. (Disk Jockey will terminate if it cannot create the file.)

You communicate with Disk Jockey by using the CLI CONTROL command. You must have write access to the directory where the IPC file is created. If you do not respond to the Disk Jockey prompt in 60 seconds, Disk Jockey will redisplay the prompt.

The general IPC message format is:

```
From Pid xxxxx: (DJ) hh:mm:ss
From Pid xxxxx: (DJ)
<messages>
To respond: CONTROL <IPC file pathname> <answer>
```

For example, user John starts a batch job running the script file MY_LDCOPY_SCRIPT with the /WAIT switch:

This is what John enters at his terminal:

```
) DIR  :UDD:JOHN ↵
) QBATCH  DJ/SCRIPT=MY_LDCOPY_SCRIPT/WAIT ↵
...
```

This is what Disk Jockey displays on the system console, and the CONTROL command line that John types:

```
From Pid 00053: (DJ)  10:03:23
From Pid 00053: (DJ)
Hard tape error
Retry (R), Skip block (S), or Abort (A)? [S]
To respond: CONTROL :UDD:JOHN:DJ00053 <answer>

) CONTROL  :UDD:JOHN:DJ00053  R ↵
```

Troubleshooting a Script File

Usually, when an error occurs, Disk Jockey displays an error message on the user's screen and writes the message to the log file. You can therefore use the CLI command TYPE or the BROWSE utility to display the log file if there is a problem. If the log file does not exist, it is probably because Disk Jockey does not have write access to the directory. Ensure proper ACL is set for the directory (username,WARE).

When you run Disk Jockey in a batch stream and Disk Jockey encounters a problem, Disk Jockey will prompt PID 2 or whomever /MESSAGE=pid is set to to resolve the problem. If the pid no longer exists, Disk Jockey will terminate and write an error message to the log file. If the pid belongs to a different user since the batch job was submitted, Disk Jockey will be unable to differentiate the intended user and the current user by pid number and will continue as normal.

Figure 7–8 shows an example of a log file.

```
Script file: :UTIL:MY_SCRIPT
Date: 04-Aug-1993
Time:  9:26:28

******************** [LDCOPY] ********************

Verifying section ...

Source       => Unit: DPJ1
                LDU Name: DISK1
                LDU ID: DISK1
Destination  => Unit: MTJ0

                 ** Mount Request **
Tape unit: MTJ0,  Tape set sequence number: 0001
Volume ID: LDC001
Press NEW LINE when ready

  Label              Expected            Found

Volume ID:           LDC001              LDC001
Filename:            LDCOPY BACKUP       LDCOPY BACKUP
Sequence number:     0001                0001
Generation:          0001                0001
Creation date:       9/28/92             9/28/92
Expiration date:     12/27/92            12/27/92

*** Tape retention period has not expired  ***
Correct and continue (C), Abort (A), or Ignore (I)? [C]

*** Buffer size not supported by tape unit ***
*** Tape buffer adjusted to 16384          ***

Disk to Tape copy in progress...
Copy has completed successfully

******************** End of Section ********************
```

Figure 7–8 An Example of an LDCOPY Log File

End of Chapter

Chapter 8
Using MSCOPY to Back Up and Restore AOS/VS Files

Read this chapter when

- You want to copy only those data sectors of an AOS/VS file modified since the last backup;
- You want to save time in copying large files, such as databases.

MSCOPY is an AOS/VS–only utilityI. MSCOPY offers an alternative to file–oriented backups (with DUMP/DUMP_II) and entire LDU backups (with PCOPY). MSCOPY does a form of incremental backup in which only disk sectors (blocks) that have changed since the last full backup are copied. MSCOPY works only with disk units that can keep track of modified sectors: Model 6236 (354 Mbyte), Model 6239 (592 Mbyte), and Model 6297 (862 Mbyte), DPJ–type units.

You might consider using MSCOPY if

- you have a nonsystem LDU composed of Model 6236 or 6239 disks that has one or more large database files, and the file(s) change relatively little between backups; and
- this LDU can be released from the system for thirty minutes or so each day; and
- you're willing to use DUMP/DUMP_II or PCOPY to back up the system LDU.

MSCOPY backs up and recovers modified disk sectors (disk blocks). It can copy only modified sectors: those that have changed since the last full MSCOPY backup. MSCOPY is most useful for sites that have very large files where relatively few changes occur (like very large INFOS II or DG/DBMS database files). It takes a long time to back up these files with DUMP/DUMP_II or PCOPY; it takes less time to back up only the changed sectors with MSCOPY.

MSCOPY does not work well in conjunction with DUMP/DUMP_II or PCOPY. MSCOPY runs while AOS/VS is up, and since MSCOPY can't copy an open LDU, it can't copy the system LDU. So, if you use MSCOPY to copy nonsystem LDU(s), you will still need to use need DUMP/DUMP_II or PCOPY to back up the system LDU.

DUMP/DUMP_II and PCOPY can both be used for backup; for example, you can run PCOPY once a week and do incremental backups on the intervening days with DUMP/DUMP_II.

How MSCOPY Works

When MSCOPY does a full backup of an LDU, it creates a backup history file for that LDU. The history file records information (like date and filename) on the full backup and all subsequent incremental backups, until the next full backup.

After the full backup, the LDU is reinitialized into the system and the disk sectors are modified. When any sector is modified, its modified sector bit is set and stays set until the next full backup. Each incremental backup copies all sectors modified since the last incremental backup.

As time passes and the number of modified sectors grows, so does the size of the incremental backups. Eventually, the incremental backups grow large enough to make another full backup worthwhile.

When you do another full backup, MSCOPY renames the last history file and starts a new history file. After rewinding the last full backup tape, MSCOPY zeroes all modified sector bits on the LDU, indicating that all sectors are unmodified.

To restore an LDU using MSCOPY, you need restore only the most recent full backup and most recent incremental backup.

The backup history file allows MSCOPY to keep track of all tapes in a backup tape set. (When a backup — either full or incremental — occupies more than a single volume of tape, the total of all the tapes is referred to as a tape set.) If you restore a set, MSCOPY will use the pertinent history file to ensure that you restore both the full backup and the last incremental backup. All backup history files are stored in the directory :UTIL:MSCOPY_HISTORY; away from the copied LDU. The MSCOPY program creates this directory the first time you run it. MSCOPY expects to find the history files in :UTIL:MSCOPY_HISTORY; don't delete or move this directory.

The MSCOPY utility is composed of several different files. The AOS/VS system tape includes the following MSCOPY files in the following directories:

Directory	File	Contents
:UTIL	MSCOPY.CLI	Macro that executes the MSCOPY program
:UTIL	MSCOPY.PR	The MSCOPY program
:HELP	CLI.TPC.MSCOPY	Help files
:UTIL:MSCOPY	MSCOPY.BTSFPLK2	Error messages

MSCOPY Menu Options

MSCOPY offers the following options on its menu.

1 NEW	Does a full backup and creates a new history file. You must use NEW the first time you run MSCOPY on an LDU. You must use FULL and NEXT thereafter.
2 FULL	Does a full backup from an existing LDU, renames the old history file, and creates a new history file. Use FULL whenever you want a full backup.
3 NEXT	Does an incremental backup that includes all modified sectors. NEXT is the MSCOPY option you'll use most often. Part of the display tells you what percentage of the LDU has been copied; use this information to decide when it's time for a full backup.
4 RESTORE	Restores an LDU from a backup tape.
5 HELP	Gives help on MSCOPY commands.
6 HISTORY	Displays dates and other information in a history file, including the percentage of the LDU copied on each backup.
7 BYE	Exits from the MSCOPY program.

MSCOPY Command Line and Switches

The MSCOPY command line is

MSCOPY *[switch]*

The switches are

/1= *IGNORE* *WARNING* *ERROR* *ABORT*
This sets CLASS1 (serious) error handling to the level you specify. The default is ERROR. We suggest you retain the default (omit this switch). For more detail on these levels, see the manual *Using the CLI (AOS/VS and AOS/VS II)*.

/2= *IGNORE* *WARNING* *ERROR* *ABORT*
This sets CLASS2 (mild) error handling to the level you specify. The default is WARNING. We suggest you retain the default (omit this switch).

/DENSITY= *800* *1600* *6250*
For backup only. This switch sets the recording density, in bits per inch (b/in). It overrides the setting set by the CPU's front panel DENSITY switch, if any. If you omit /DENSITY, the density of the panel switch is used. If the unit has no switch, the density at which the last tape was written is used. Generally, you'll want to use the highest density the unit provides. A density of 6250 b/in is available only on MTD units; 800 b/in is available only on MTB units. If you plan to use multiple tape units, specify a density available on all units.

/E=pathname
Copies MSCOPY error messages to the file named in pathname, creating the file if it doesn't exist. Error messages are also sent to the console and listing file, if any.

/L=pathname
Copies MSCOPY dialog to the file named in pathname. If the file doesn't exist, MSCOPY creates it; if it does exist, MSCOPY appends to it. This switch is useful when you want a hardcopy of an MSCOPY session.

/RETAIN=days
By default, if you omit this switch, MSCOPY sets the tape expiration date to 90 days after the current date. If anyone tries to do an MSCOPY backup to an unexpired tape, MSCOPY will display

Tape has not exceeded retention period.
MOUNTED new reel – try again
RELABEL tape and continue
Select action (MOUNTED or RELABEL) [MOUNTED]

And you will need either to type RELABEL and press NEW LINE or find a volume without an unexpired label, mount it, and type MOUNTED (and press NEW LINE). MSCOPY will ask this question for every unexpired volume, which can be a nuisance for a full backup of 15 volumes or more.

For maximum flexibility, you might want to specify /RETAIN=0, which marks the tape as already expired, allowing it to be reused immediately.

On the other hand, a nonzero retention period can be useful. It can help prevent backups you want to keep from being overwritten. To make good use of the retention period, you must decide how long you want to keep your tape sets, and you must acquire enough tape for all the backup sets needed. Also, you will need to be consistent: specify the same retention period for each backup set.

If you decide on a retention period other than 90 days, specify it with /RETAIN=days. You might want to edit the MSCOPY.CLI macro to specify your retention period. If 90 days is sufficient, omit the /RETAIN switch.

/STATISTICS Tells MSCOPY to display statistics: number of tape blocks and disk sectors read or written, bad block count, and so on. This switch is primarily useful for DG program developers.

/VERIFY For backup only. MSCOPY can verify the tapes that it wrote by comparing their data to the LDU data. If all tapes verify without errors, you can be nearly certain that they will restore properly later. (Bad storage conditions can make tapes deteriorate after you verify them.) Verifying takes about as long as backing up.

When you verify, MSCOPY reads the tapes in the same order that it wrote them. So, for a multiple–tape backup, you will need to dismount the last tape(s) and mount the first tape(s). MSCOPY will then read the first tape label and prompt for other tapes as needed.

Usually, you can judge tape condition by the number of soft tape errors reported on the system console, and use this as a basis for replacing tapes. If you must take every step to ensure that the backup can be restored, include this switch.

If You Make a Mistake with MSCOPY

If you make a typing mistake, before pressing NEW LINE, press the DEL key to erase characters one by one; or enter CTRL–U to erase the entire line.

If you've already pressed NEW LINE after a wrong answer, MSCOPY will probably recognize your error and repeat the question. If so, type the correct answer. MSCOPY error messages are explained in the manual *AOS/VS and AOS/VS II Error and Status Messages*.

If you decide to abort MSCOPY, type CTRL–C CTRL–B or press the BREAK/ESC key (ESC on some terminals) and type ABORT (and press NEW LINE). Then repeat the original MSCOPY command line.

If you abort MSCOPY during an incremental backup, full backup, or restore, the copy will be invalid. You must restart it from the beginning.

You can get help (a list of MSCOPY commands) by pressing the BREAK/ESC key or typing CTRL–C CTRL–A, and then typing HELP and pressing NEW LINE. To learn the dates of MSCOPY backups, choose option 6, "HISTORY" from the MSCOPY Main Menu.

Running MSCOPY

You need Superuser privilege to access the history file and disk unit(s) that make up the LDU. You must use an upper– and lowercase terminal, because MSCOPY dialog is upper– and lowercase.

Any time during a backup or restore operation, you can check progress with the STATUS command. Press the BREAK/ESC key (don't press CMD first); then type STATUS and press NEW LINE. Press BREAK/ESC only while MSCOPY is doing I/O. If you press BREAK/ESC while MSCOPY is asking a question, it has the same effect as pressing the NEW LINE key (it tells MSCOPY to use the default answer).

By default, MSCOPY requires operator action after it uses each tape. On a backup, though, you can premount additional tapes after specifying the first tape. To do this, press the BREAK/ESC key; then type the premount command for each other tape unit, explained below. MSCOPY will then access the units in sequence, without operator interaction.

1. When you plan a full backup, make sure you have enough tapes and time. The number of tapes and minutes varies with the amount of space used on the LDU. A 2,400–foot tape, at 1,600 b/in, can hold about 38 Mbytes (74,000 disk blocks). It takes 10–12 minutes to fill if system load is light. A Model 6236 disk unit can hold 354 Mbytes; a Model 6239 disk can hold 592 Mbytes.

   When an LDU is initialized, you can type SPACE ldu–name and press NEW LINE to tell how many disk blocks are being used (the CUR figure).

   On a full (and new) backup, after MSCOPY has copied the LDU to tape and rewinds the last tape, it requires up to 20 minutes per disk (depending on how full the disk is) to zero all the modified bits on the LDU.

   When you use MSCOPY to copy or restore an LDU, all disk units must be turned on and ready. The LDU can't be initialized (release it before starting MSCOPY if it is initialized). MSCOPY will run faster if system load is light.

2. Start MSCOPY by typing

) MSCOPY ↵
   MSCOPY will respond by displaying its menu.

   MSCOPY Revision n

   1 NEW – Perform a full backup, and create a new backup set.
   2 FULL – Perform a full backup, and supersede the previous backup set.
   3 NEXT – Perform an incremental backup, and add to the currrent backup set.
   4 RESTORE – Restore an LDU from a backup set.
   5 HISTORY – Display a backup set's history.
   6 HELP – Display help file.
   7 BYE – Leave MSCOPY.

   Please select action []

3. To select the option you want, you can type either the number of the menu option (for example, 1 and press NEW LINE) or specify the option word (type the word NEW and then press NEW LINE).

   Option 1, "NEW," is required for backup if you've never run MSCOPY on the LDU before. Option 2, "FULL," does a full backup and starts a new backup set. Option 3, "NEXT," does an incremental backup, based on modified sectors.

   Option 4, "RESTORE," starts restoring, based on a backup set's full and last incremental backup. Usually, you'll want to restore the most recent backup tape set, although you can restore any backup set. Option 5, "HISTORY," displays the dates and labeled tape filenames of all backups in any backup set.

   Option 6, "HELP," explains MSCOPY commands; and option 7, "BYE," exits to the CLI.

   For example, for a full backup that is to start a new backup set, you'd select option 2, "FULL," or type the word FULL and press NEW LINE.

4. With any option but option 6, "HELP," or option 7, "BYE," you need to specify the LDU name.

   Enter name of object Logical Disk Unit (LDU) []
   >

   Type the name of the LDU you want to back up, restore, or check the backup history of. This must be the LDU name as assigned by the Disk Formatter. For example

   > DATABASE ↵ (Type LDU name next to the MSCOPY (>) prompt.)

5. The following question is asked if you chose option 1, "NEW," or option 4, "RESTORE"

   Enter names of all disk units in LDU [default]
   >

   For the new backup set, you must identify disk units. MSCOPY runs only on DPJ–type disks, Models 6236 and 6239. Type the unit names of all disks in the LDU. Omit the leading @. For example, if the LDU includes the second and third units on the first controller:

   > DPJ1 DPJ2 ↵ (Type unit names next to the MSCOPY (>) prompt.)

   When you restore, the default value (displayed in square brackets) shows disk unit(s) in the original LDU. To restore to the same unit(s) that were backed up, press NEW LINE to accept the default disk unit names shown.

   On the other hand, to restore to different disk unit(s), type the desired disk unit name(s). You might want this if one of the original units were down, and you had an identical disk (formatted the same way) in a different unit. For example, assume you have three units: DPJ0, DPJ1, and DPJ2. You need the LDU that normally runs in DJP1, but unit DPJ1 is down. You can get along without DPJ2. You have backup sets for all units.

If the disk in DPJ2 is formatted to be identical to DPJ1, you can tell MSCOPY to restore the DPJ2 backup to DPJ1 by specifying

> DPJ2 ↵

Later, when DPJ1 is fixed, you would probably restore the DPJ1 backup again, this time to DPJ1, and restore the DPJ2 backup to DPJ2.

6. The following question is asked only when you select option 4, "RESTORE," or option 5, "HISTORY."

Enter name of backup history file [MS_HISTORY.ldu–name]

Nearly always, when you want to restore an LDU, you want the most recent backup. This is also true for the history file. Information on the most recent backup is kept in the file MS_HISTORY.ldu–name. To restore the most recent backup, or check the most recent history file, press NEW LINE.

If you want to restore or check a backup history file that isn't the most recent, type the name of history file for the desired backup set, and then press NEW LINE.

Old backup history filenames have the form MS_HISTORY.ldu–name.yymmdd (the year, month and day when the file was renamed and a new history file was created.) For example, file MS_HISTORY.DATABASE.890901 contains information on the backup set for the LDU named DATABASE; started with a full backup on September 1, 1989.

Any backup set you restore will overwrite any material on the LDU. It's a good idea to make sure you have all the tapes of a backup set available before starting a restoration.

For option 5, "HISTORY," MSCOPY displays the history file and then redisplays the MSCOPY menu. You can proceed with backup or restore operations, or you can leave MSCOPY (option 7, "BYE").

7. For any backup or restoration, MSCOPY asks you to mount tape. When doing a backup, the message you will see is

Tape set filename is x.nnn.yymmdd.hhmm
Mount reel 1, ready tape unit,
and enter tape unit name [MTB0]

When doing a restoration, the message you will see is

Mount reel 1 of tape file x.nnn.yymmdd.hhmm, ready tape unit,
and enter tape unit name [MTB0].

The filename begins with a single check character (x), assigned by the operating system, and based on the time of day. This is followed by a three–digit number that shows the sequence in the backup set (nnn), the date (yymmdd), and the time (hhmm), based on a 24–hour clock.

E.000.890325.1723	(This tape filename indicates a full backup. The sequence is 000. The year is 89, the month is 03 (March, the day is the 25th, and the time is 5:23 p.m. (1723 on a 24–hour clock).)
D.001.890327.1744	(This filename indicates a first incremental backup. The sequence is 001. The date is March 27, 1989; the time is 5:44 p.m.)
J.013.890415.1803	(This filename belongs to a 13th incremental backup. The sequence is 013. The date is April 15, 1989; the time is 6:03 p.m.)

On a backup, the tape set filename should be written somewhere on a paper label on the first backup tape. The name is a useful identifier for the tape set and its sequence in the backup set. Having it visible will be handy if you need to restore an LDU.

If you're restoring, MSCOPY expects you to mount the tape set with the filename displayed.

Make sure the tape you want written to (for a backup) or read from (for a restore) is mounted, and that the tape unit is ready. Then type the unit name, without the leading @; or, to accept the default tape unit, MTB0, press NEW LINE.

8. MSCOPY now checks the tape unit to make sure it's ready. Then it begins the backup or restoration. Each 2,400–foot tape, at 1,600 b/in, takes 10–12 minutes to write or read. If MSCOPY encounters an error condition, it displays an error message. MSCOPY error messages are documented in the manual *AOS/VS and AOS/VS II Error and Status Messages*.

9. On a backup, you can premount subsequent tapes so that MSCOPY can continue without operator intervention.

   If you want to premount, do it now, while MSCOPY is writing to the tape. (If you don't want to premount other tapes, or if you are doing a restore, skip to step 10.)

   You can premount as many tapes as you have available tape units. MSCOPY will use all the reels it needs, and, if it needs more volumes, prompt the operator to mount them.

   If the tape units have density switches, switch to HIGH DENSITY. Make the unit(s) ready. For backup, the order of tapes isn't critical.

   After mounting the tapes, press the BREAK/ESC key to enter command mode. At the command mode prompt (*>), use the following format:

   PREMOUNT tape–unitname

   MSCOPY will check the tape on the unit you specify with the PREMOUNT command, verify the premount (or display an error message), and redisplay the command mode prompt. Type as many premount commands as you want; then press NEW LINE to leave command mode.

A sample backup tape sequence with two tape units, MTB0 and MTB1, looks like this:

... (Mount tapes on units) ...

Tape set filename is C.000.890720.1750
Mount reel 1, ready tape unit, and enter tape unit name [MTB0]
> ↵ (Press NEW LINE to select the default tape unit for the first reel.)

> ESC (Press the BREAK/ESC key to enter the command mode.)

*> PREMOUNT MTB1 ↵
Unit MTB1 premounted (MSCOPY verifies premount.)
*> ↵ (Pressing NEW LINE terminates command mode.)

>

... (MSCOPY writes to tape on unit 0, then displays *Unit premounted* message, and begins writing to the tape on unit 1) ...

Tape set filename is C.000.890720.1750
Mount reel 3, ready tape unit, and enter tape unit name [MTB0]
>

... (Dismount the newly written tapes; mount two more reels) ...

> ↵ (Again, select default unit named for the first reel.)

> ESC (Press the BREAK/ESC key to enter command mode.)

*> PREMOUNT MTB1 ↵
Unit MTB1 premounted (MSCOPY verifies)
*> ↵
>

MSCOPY writes to the tape on unit 0, then to the tape on unit 1. This two–reel sequence repeats until the full backup is done.

Premounting tapes is desirable because it lets MSCOPY handle multiple tapes without operator intervention, allowing you to leave the console for periods longer than 12 minutes.

10. MSCOPY will spin tape until it has backed up or restored the entire LDU. When MSCOPY finishes the current reel(s), it asks

Tape set filename is x.nnn.yymmdd.hhmm
Mount reel n, ready tape unit, and enter tape unit name [MTB0]

For a restore, the message is

Mount reel n of tape file x.nnn.yymmdd.hhmm, ready tape unit, and enter tape unit name [MTB0]

Dismount the mounted reel(s), mount the next reel(s), type the tape unit name (or press NEW LINE to accept the default tape unit name, MTB0). Then return to step 8 for the tape I/O.

On a new or full backup, after MSCOPY writes to the last tape, it zeroes the modified sector bits on the LDU. This takes up to 20 minutes per disk, depending on how full the disk is.

11. At the end of a new or full backup, MSCOPY says

New history file MS_HISTORY.ldu–name created

and for a full backup it adds:

Previous history file is MS_HISTORY.ldu–name.yymmdd

The backup history filename is based on the LDU name. The current backup history file (for the most recent full backup) is MS_HISTORY.ldu–name.

For a new LDU, MSCOPY does a full backup and creates the backup history file. (MSCOPY checks for the backup history file at the beginning, and won't let you do a new backup if the history file already exists.)

For a full backup of an old LDU, MSCOPY does a full backup and renames the old history file to MS_HISTORY.ldu–name.yymmdd. Before renaming the old file, MSCOPY deletes any existing file of the same name. Then, MSCOPY creates the new backup history file.

This arrangement makes it easy to identify backup history files. A MS_HISTORY filename that ends with characters like .890523 (for example, MS_HISTORY.DATABASE.890523) is an old history file. A MS_HISTORY file that ends in the LDU name is a current history file.

The naming arrangement means you can't do a new backup when a history file exists, and that you can't retain two incremental or full backup sets for the same day (a minor restriction) unless you manually rename the first backup history file from the CLI before the second backup.

12. When MSCOPY is done, it returns to the CLI.

Dismount all tapes, making sure they have paper labels with the full dump date, incremental dump date, and sequence number. Store the tapes safely, without write–enable rings. Tape storage suggestions appear early in this chapter.

MSCOPY Backup Examples

This section shows you examples of a new (or first) full backup, the first incremental backup, the nth incremental backup, and finally, the next full backup of an AOS/VS LDU using MSCOPY. The next section will show you how to restore an LDU using MSCOPY.

The New (First) Full Backup

June 20, 1989. Computer site decides to start using MSCOPY to copy an LDU named DATABASE that includes disk units DPJ1 and DPJ2. Tape units will be MTB0 and MTB1, with MTB1 premounted.

```
) MSCOPY ↵

1 NEW – Perform a full backup, and create a new backup set.
2 FULL – Perform a full backup, and supersede the previous backup set.
3 NEXT – Perform an incremental backup, and add to the current backup set.
4 RESTORE – Restore an LDU from a backup set.
5 HISTORY – Display a backup set's history.
6 HELP – Display help file.
7 BYE – Leave MSCOPY.

Please select action []
> 1 ↵

Enter name of object Logical Disk Unit (LDU) []
> DATABASE ↵

Enter names (separated by spaces) of all disk units in LDU []
> DPJ1 DPJ2 ↵

Tape set filename is C.000.890620.1750
Mount reel 1, ready tape unit, and enter tape unit name [MTB0]
                                  (Operator mounts the first tape on unit MTB0,
                                  and the second tape on unit MTB1.

> ↵                               (Operator then presses NEW LINE to accept the
                                  default tape unit, MTB0.)
>

....(Tape I/O starts) ...

ESC                               (Operator presses BREAK/ESC key to enter
                                  command mode.)
                                  (Operator premounts second tape on unit MTB1,
                                  then returns to the system console and types...)

*> PREMOUNT MTB1 ↵
```

Figure 8–1 MSCOPY New Backup Example (continued)

Unit MTB1 premounted	(MSCOPY verifies that a tape is premounted on unit MTB1.)
*> ↵	(Press NEW LINE to exit from command mode.)
>	

... (Backup proceeds, first dumping to the tape on unit MTB0; then to the tape on MTB1) ....

Tape set filename is C.000.890620.1750
Mount reel 3, ready tape unit, and enter tape unit name [MTB0]

... (Operator dismounts two newly written tapes, and mounts two new tapes on units MTB0 and MTB1.) ...

> ↵	(Operator selects unit MTB0 by pressing NEW LINE.)
ESC	(Operator presses BREAK/ESC key to enter command mode.)
*> PREMOUNT MTB1 ↵	(Operator tells MSCOPY that a tape has been premounted on unit MTB1.)
Unit MTB1 premounted	(Press NEW LINE. Then repeat the original MSCOPY command line. MSCOPY verifies that a tape has been premounted.)
*> ↵	(Operator presses NEW LINE to exit command mode.)
>	

... (Backup proceeds, first writing to the tape on MTB0; then to the tape on MTB1.) ...

... (Backup ends, and MSCOPY rewinds last tape used. There is a delay while MSCOPY zeros the modified bits.) ...

New history file MS_HISTORY.DATABASE created

)	(MSCOPY returns you to the CLI when it finishes.)

(When the full backup finishes, the operator dismounts all tapes and stores them safely.)

Figure 8–1 MSCOPY New Backup Example (concluded)

The First Incremental Backup

June 21 — the next workday. 7:00 p.m. arrives, time for the incremental backup. Only one tape unit will be used, since the operator expects the incremental backup to fit on one reel.

```
) MSCOPY ↵

MSCOPY Revision n

1 NEW - Perform a full backup, and create a new backup set.
2 FULL - Perform a full backup, and supersede the previous backup set.
3 NEXT - Perform an incremental backup, and add to the current backup set.
4 RESTORE - Restore an LDU from a backup set.
5 HISTORY - Display a backup set's history.
6 HELP - Display help file.
7 BYE - Leave MSCOPY.

Please select action []
> 3 ↵

Enter name of object Logical Disk Unit (LDU) []
> DATABASE ↵

Tape set filename is T.001.890621.1905
Mount reel 1, ready tape unit, and enter tape unit name [MTB0]

(Operator mounts a tape on tape unit MTB0.)

> ↵                          (Pressing NEW LINE accepts the default tape
                             unit name given in brackets.)

... (Incremental backup proceeds) ...

)                            (When MSCOPY finishes, it returns you to the
                             CLI prompt.)
```

When the incremental backup finishes, the operator dismounts the tape and stores it safely.

Figure 8–2 MSCOPY First Incremental Backup Example

The nth Incremental Backup

June 29 — After several days of incremental backups, the number of modified sectors has grown; requiring more backup tape.

```
) MSCOPY ↵

MSCOPY Revision n

1 NEW - Perform a full backup, and create a new backup set.
2 FULL - Perform a full backup, and supersede the previous backup set.
3 NEXT - Perform an incremental backup, and add to the current backup set.
4 RESTORE - Restore an LDU from a backup set.
5 HISTORY - Display a backup set's history.
6 HELP - Display help file.
7 BYE - Leave MSCOPY.

Please select action []
> 3 ↵

Enter name of object Logical Disk Unit (LDU) []
> DATABASE ↵

Tape set filename is G.029.890629.1859
Mount reel 1, ready tape unit, and enter tape unit name [MTB0]

... (Operator mounts a tape on tape unit 0.)

> ↵                          (Pressing NEW LINE accepts the default tape
                             unit name given in brackets.)

... (Incremental backup proceeds) ...

Tape set filename is G.029.890629.1859
Mount reel 2, ready tape unit, and enter tape unit name [MTB0]

... (This incremental backup requires a second tape. The operator dismounts the
first tape and mounts the second tape on MTB0.)...

> ↵                          (Pressing NEW LINE accepts the default tape
                             unit name given in brackets.)

... (Incremental backup continues to completion.) ...

)                            When MSCOPY finishes, it returns you to the
                             CLI prompt.)

When the incremental backup finishes, the operator dismounts the tape and
stores it safely.
```

Figure 8–3 MSCOPY Next Incremental Backup Example

The Next Full Backup

June 30 — Since the last incremental backup required more than one tape, the operator decides to do a full backup. Since the operator knows that this full backup will use at least two tapes, he/she will use two tape units; premounting the second tape on tape unit MTB1.

```
) MSCOPY ↵

MSCOPY Revision n

1 NEW - Perform a full backup, and create a new backup set.
2 FULL - Perform a full backup, and supersede the previous backup set.
3 NEXT - Perform an incremental backup, and add to the current backup set.
4 RESTORE - Restore an LDU from a backup set.
5 HISTORY - Display a backup set's history.
6 HELP - Display help file.
7 BYE - Leave MSCOPY.

Please select action []
> 2 ↵

Enter name of object Logical Disk Unit (LDU) []
> DATABASE ↵

Tape set filename is U.000.890630.1922
Mount reel 1, ready tape unit, and enter tape unit name [MTB0]

... (Operator mounts the first tape on tape unit MTB0; the second tape on tape unit
MTB1.)...

> ↵                              (Pressing NEW LINE accepts the default tape unit
                                  name given in brackets.)
>

....(Tape I/O starts) ...

ESC                              (Press the BREAK/ESC key to enter
                                  command mode.)

*> PREMOUNT  MTB1 ↵              (Premount tape on unit MTB1.)
Unit MTB1 premounted             (MSCOPY verifies premount.)

*> ↵                             (Press NEW LINE to exit command mode.)
>

... (Backup proceeds, first writing to the tape on MTB0; then the tape on MTB1.)...

Tape set filename is U.000.890630.1922
Mount reel 3, ready tape unit, and enter tape unit name [MTB0]

... (This full backup requires additional reels of tape. The operator dismounts the
first and second reels, and mounts the third and fourth reels.) ...
```

Figure 8–4 MSCOPY Full Backup Example (continued)

> ↵	(Pressing NEW LINE accepts the default tape unit name given in brackets.)
ESC	(Press the BREAK/ESC key to enter command mode.)
*> PREMOUNT MTB1 ↵	(Premount tape on unit MTB1.)
Unit MTB1 premounted	(MSCOPY verifies.)
*> ↵	(Pressing NEW LINE exits command mode.)
>	

... (Backup continues, first writing to the tape on MTB0; then the tape on MTB1.) ...

... (Backup ends, MSCOPY rewinds last tape used. Delay occurs while it zeros the modified bits.) ...

Backup history file MS_HISTORY.DATABASE created
Previous history file renamed to MS_HISTORY.DATABASE.890620

)

When the incremental backup finishes, the operator dismounts all the tapes, and stores them safely.

Figure 8–4 MSCOPY Full Backup Example (concluded)

Restoring an LDU with MSCOPY

When you restore an LDU with MSCOPY, you restore the entire LDU. To restore an AOS/VS LDU using MSCOPY, follow these steps.

1. If the disk(s) has not been formatted, run a Full format on the disk with the Disk Formatter utility. Specify the same LDU ID and name as the original LDU had. You must specify (or default) the same addresses for the bitmap, overlay area (if any) and remap area that were given for the original LDU.

2. Get the desired set of backup tapes. Generally, you'll want the most recent set (described by backup history file MS_HISTORY.ldu–name in directory :UTIL:MSCOPY). But if you want a different backup set, you can check MS_HISTORY+ files in directory :UTIL:MSCOPY_HISTORY. Files with a six–digit suffix are old files; the suffix indicates the full backup date (yymmdd).

   When you restore an LDU, MSCOPY must read tapes in the correct order. It will allow you to change a tape if you mount the wrong one.

3. Start MSCOPY from a user terminal or the system console. Restore the full backup first. On a restoration, you must specify the tapes one by one; you can't premount (as you can when backing up an LDU using MSCOPY).

   Then restore the most recent (latest) incremental backup, unless you want to restore the LDU to a state before the most recent incremental backup. MSCOPY will consult the backup history file and warn you if the backup is not the most recent one. It will allow you to restore a backup that's not the most recent.

4. If you want AOS/VS to run from the restored LDU, you must install a disk bootstrap on this LDU. (MSCOPY does not install this bootstrap.) Run the Installer as follows:

   ```
   ) SUPERUSER ON ⤵
   Su) XEQ :UTIL:INSTL ⤵

   AOS/VS Installer Rev n
   Specify each disk in the LDU
   Disk unit name? DPJn ⤵                (Specify all units in restored LDU.)

   — Disk bootstrap installed
   Do you want to install a System Bootstrap [Y] ? ⤵

   Do you want to install a System [Y] ? N ⤵
   Done

   Su) SUPERUSER OFF ⤵
   )
   ```

   Now, AOS/VS should run from the restored LDU.

5. Put the covers on all tapes and store tapes safely.

6. You're done! You've restored the entire LDU — with luck, losing only a little work (the changes made since the last backup occurred). The next MSCOPY backup can be a full or incremental backup.

Restoration Example

(March 20, 1989. A computer site decides to restore the most recent backup set for its LDU named DATABASE. The LDU spans disk units DPJ1 and DPJ2.)

) MSCOPY ↵

MSCOPY Revision n

1 NEW – Perform a full backup, and create a new backup set.
2 FULL – Perform a full backup, and supersede the previous backup set.
3 NEXT – Perform an incremental backup, and add to the current backup set.
4 RESTORE – Restore an LDU from a backup set.
5 HISTORY – Display a backup set's history.
6 HELP – Display help file.
7 BYE – Leave MSCOPY.

Please select action []
> 4 ↵

Enter name of object Logical Disk Unit (LDU) []
> DATABASE ↵

Enter name of backup history file [MS_HISTORY.DATABASE] ↵
Enter names of all disk units in LDU [DPJ1 DPJ2] ↵

Mount reel 1 of tape file V.000.890302.1913, ready tape unit, and enter tape unit name [MTB0]

... (Operator mounts first tape on MTB0, and presses NEW LINE.) ...

> ↵ (Pressing NEW LINE accepts the default tape unit name; MTB0.)

... (Restoration proceeds. MSCOPY reads the first tape.) ...

Mount reel 2 of tape file V.000.890302.1913, ready tape unit, and enter tape unit name [MTB0]

... (Operator dismounts first tape; mounts second tape on MTB0.) ...

> ↵ (Pressing NEW LINE accepts default tape unit name; MTB0.)

... (Restoration continues; MSCOPY reading from the second tape.) ...

Mount reel 3 of tape file V.000.890302.1913, ready tape unit, and enter tape unit name [MTB0]

... Operator dismounts second tape; mounts third tape on MTB0.) ...

> ↵ (Pressing NEW LINE accepts default tape unit name; MTB0.)

... (Restoration continues; MSCOPY reading from the third tape.) ...

Mount reel 4 of tape file V.000.890302.1913, ready tape unit, and enter tape unit name [MTB0]

... (Operator dismounts third tape; mounts fourth tape on MTB0.) ...

> ↵ (Pressing NEW LINE accepts default tape unit name; MTB0.)

... (Restoration proceeds) ...

... (Operator continues to dismounts tapes when finished, and mount next one, until all the backup tapes have been read.) ...

Mount reel 1 of tape file Q.006.890319.1908, ready tape unit, and enter tape unit name [MTB0]

... (MSCOPY has prompted for the first tape in the last incremental set. You can override this by mounting an earlier incremental from the set and telling MSCOPY to continue after the error message. But here, the operator mounts the volume MSCOPY asks for on MTB0, and mounts the second volume on MTB1.) ...

> ↵ (Pressing NEW LINE accepts default tape unit name; MTB0).

... (Restoration continues) ...

Mount reel 2 of tape file Q.006.890319.1908, ready tape unit, and enter tape unit name [MTB0]

> MTB1 ↵ (Instructs MSCOPY to read from the tape on unit MTB1.)

... (Restoration concludes) ...

) (When finished, MSCOPY returns you to the CLI prompt.)

(The operator stores the full and incremental backup tapes; then initializes and checks the LDU.)

End of Chapter

Chapter 9
Using PCOPY to Back Up and Restore AOS/VS LDUs

Read this chapter

- When you want to make a physical copy of AOS/VS files;
- When you want to back up or restore entire AOS/VS LDUs.

PCOPY works only on AOS/VS files, and copies all occupied space on an LDU. There are two versions of PCOPY: a stand–alone version, which can copy to and from any LDU; and a stand–among version, which can copy to and from any uninitialized LDU (this excludes the master LDU, which is initialized when AOS/VS is running.

You can start stand–alone PCOPY directly from disk, as if it were an operating system. It is in the root directory; filename PCOPY.PR. You can start stand–among PCOPY from the CLI (via XEQ); it is file PCOPY.PR in directory :UTIL. To run the stand–among PCOPY, you must have the following privileges in your user profile: *Change type* and *Change address space* type.

PCOPY is useful if you must use cartridge tapes for backup, since it can work much faster than DUMP_II with these tapes.

PCOPY is ideal for disk–to–disk backup if you have two or more identical disk units with removable disks, and LDUs of one physical disk. It is also useful for disk–to–disk backup if you have three or more identical disk units with removable packs. PCOPY can also copy to and from diskette.

You might consider using PCOPY to backup and restore files if

- You prefer to do only full backups of entire LDUs;
- You must use cartridge tape for backups;
- You have two identical disk units, with removable disk packs (three for a two–disk LDU); and,
- Your system can be shut down for a while each day to copy the system LDU.

PCOPY Requirements

To copy to tape or diskettes, have enough tapes or diskettes ready. A 2400–foot tape, on an MTB unit set at high density, can hold about 74,000 disk blocks (38 megabytes) of information. It takes 10–22 minutes to fill. An MTC unit, with a 1000–foot tape, can hold about 31,000 blocks (16 megabytes) and takes about 12 minutes. An MTC cartridge tape — if you use the buffer size that allows PCOPY to run the tape fast — can hold about 15,500 blocks (8 megabytes). An MTD unit, with a 2,400–foot tape, at 6250 b/in, can hold about 250,000 blocks (130 megabytes). An MTJ reel–to–reel unit, with a 1000–foot tape, can hold about 31,000 blocks (16 megabytes) and takes about 12 minutes. At a buffer size of 16 Kbytes, an MTJ cartridge tape holds 120 Mbytes (model 6352) or 20 Mbytes (model 6351). A 737,000–byte diskette can hold about 1,400 disk blocks.

Use the CLI command SPACE : and press NEW LINE to find the number of blocks used on the LDU; then divide by the pertinent number of blocks to see how many tapes or diskettes you need. The tapes/diskettes may already be labeled or you can have PCOPY label them.

When you copy to disk (from tape, diskettes, or disk), PCOPY has some restrictions. The destination LDU must be similar to the original LDU. PCOPY requires that the bitmap and remap area addresses be the same as those on the original LDUs. If the original LDU was (is) a system disk, the destination LDU must have the same overlay area address as the original. Finally, you cannot recover a disk that has a bad block in the bitmap, remap, or overlay areas.

When you copy to disk from tape or diskette, the destination LDU's unique ID should be the same as the original's. (If the IDs differ, PCOPY will issue a warning, but let you continue if you want.) When you copy from disk to disk, PCOPY assumes that LDU unique IDs are different. For any copy, PCOPY does not check the source LDU name against the destination LDU name. This means that the LDU names can differ for disk–to–disk work — but you will probably want them to be the same when you restore from tape or diskette. PCOPY error messages will tell you what to do if the LDUs differ. Usually, it's simply a matter of running the Disk Formatter on the destination LDU to move an area.

If You Make a Mistake Running PCOPY

If you make a typing error and notice it before pressing NEW LINE, press the DEL key to erase the bad characters one by one. (For stand–alone PCOPY, the system echoes an underscore for each character erased. Then type the correct characters and press NEW LINE. Or, you can type CTRL–U to delete the entire line.

If you have already pressed NEW LINE, and PCOPY has not yet written to the destination medium, you can restart it by typing CTRL–C CTRL–A when PCOPY asks the next question. If PCOPY has started writing, you can stop it only via the break sequence and RESET and NEW LINE.

If PCOPY displays an error message that you don't understand, see the PCOPY error messages in the manual *AOS/VS and AOS/VS II Error and Status and Messages*. You can restart PCOPY if it aborts by typing CONTINUE and pressing NEW LINE.

Starting PCOPY from Disk

To start the stand–alone version of PCOPY, shut down AOS/VS normally by executing the DOWN.CLI macro. After AOS/VS has been brought down, type BYE to get to the *SCP–CLI>* prompt. At the *SCP–CLI>* prompt, type BOOT and the proper device code; for example

SCP–CLI> BOOT 27 ↵ (Or the correct device code for your master LDU.)

When the Operating System Load Menu is displayed, choose option 2, "Enter the Technical Maintenance Menu."

Operating System Load Menu

1 Continue immediately with operating system load
2 Enter the Technical Maintenance Menu
. . .

Enter choice [1]: 2 ↵ (Option 2 displays the Technical Maintenance Menu.)

At the Technical Maintenance Menu, choose option 6, "Run a specified program." At the *Pathname?* prompt, type :PCPOPY and press NEW LINE; for example

Technical Maintenance Menu

. . .

6 Run a specified program
. . .

Enter choice [1]: 6 ↵

Pathname? :PCOPY ↵ (Specify PCOPY as the program you want to run.)

To start the stand–among version of PCOPY, type

) SUPERUSER ON ↵
Su) XEQ :UTIL:PCOPY ↵

When PCOPY starts up, turn to the desired "PCOPY Dialog" section in this chapter. The sections are

- Disk–to–Disk PCOPY Dialog (All Disks On Line)
- Disk–to–Disk PCOPY Dialog (All Disks Not On Line)
- Disk–to–Tape PCOPY Dialog
- Tape–to–Disk PCOPY Dialog
- Disk–to–Diskette PCOPY Dialog
- Diskette–to–Disk PCOPY Dialog

Starting PCOPY from Tape

Starting PCOPY is easier from disk. But you can also start it from tape if you choose. Only the stand–alone PCOPY runs from tape.

Before you can start PCOPY from tape, you need to make a stand–alone PCOPY tape. This is easy to do when AOS/VS is running. Mount tape, ring in, on a tape unit (for example, MTB0); put the unit on line; and type

```
) SUPERUSER ON ↵
Su) DIR : ↵
Su) COPY/V @MTB0:0 TBOOT ↵          (Use correct tape unit name.)
TBOOT

Su) COPY/V @MTB0:1 PCOPY ↵          (Use correct tape unit name.)
PCOPY

Su) REWIND @MTB0 ↵                  (Use correct tape unit name.)
Su)
```

This gives you a copy of the stand–alone PCOPY on tape. When this tape is mounted on unit 0 on the first tape controller, you can boot it and specify the tape file number you want. You can do this with any stand–alone program, if you put TBOOT on file 0 of the tape.

Before starting PCOPY, if CPU microcode isn't loaded, you must load it from the SYSTEM MEDIA tape (described in *Installing, Starting, and Stopping AOS/VS).*

To start PCOPY from your PCOPY tape, mount the tape on unit 0. Next, bring down AOS/VS in a normal fashion by executing the DOWN.CLI macro, and when the macro finishes, typing BYE to bring up the *SCP–CLI>* prompt. From the *SCP–CLI>* prompt, continue as follows:

```
SCP–CLI> BOOT 22 ↵                  (Specifies an MTB tape drive. Type 23
                                    for a MTJ–type drive or 62 for a
                                    MTD–type drive.)

Tape file number? 1 ↵               (File 1 is the PCOPY file.)
```

When PCOPY starts up, turn to the desired "PCOPY Dialog" section in this chapter.

Starting PCOPY from Diskette

Starting PCOPY is easier from disk, but you might need to start it from diskette if you can't start from disk. As with magnetic tape, the stand–alone version of PCOPY runs only from diskette. Also be aware that you must be running the 16–bit CLI in order to use PCOPY on diskettes; the 32–bit CLI does not support diskettes.

If you're building or restoring an AOS/VS system LDU, you must load the IOC emulator and microcode and format the LDU as described in the manual *Installing, Starting, and Stopping AOS/VS*. Specify the same LDU name and addresses (for example, for the bitmap) as were given for the original LDU.

Bring down AOS/VS in the normal fashion by executing the DOWN.CLI macro, and when the macro finishes, typing BYE to bring up the *SCP–CLI>* prompt. From the *SCP–CLI>* prompt, continue as follows:

At this point, AOS/VS diskette number 1 should be in unit 0. This diskette has a copy of PCOPY on it. To start PCOPY from the diskette, proceed as follows:

SCP–CLI> BOOT 64 ↵ (Device code of diskette.)

Operating System Load Menu

1 Continue immediately with operating system load
2 Enter the Technical Maintenance Menu
. . .

Enter choice [1]: 2 ↵ (Select option 2 to enter the Technical Maintenance Menu.)

Technical Maintenance Menu

. . .

6 Run a specified program
. . .

Enter choice [1]: 6 ↵ (Select option 6.)

Pathname? :PCOPY ↵ (Specify PCOPY as the program that you want to run.)

Disk–to–Disk PCOPY Dialog (All Disks On Line)

Have all disks in your source and destination LDUs ready. For a multiple–disk LDU, you can put the source LDU disks in the first group of disk units, and the destination disks in the second group of disk units. For example, the source LDU might go in units DPF0 and DPF1, and the destination LDU in units DPF10 and DPF11. (If all disks in the source and destination LDUs will not be mounted throughout the copy, go to the next section.)

As described in the "PCOPY Requirements" section, your destination disk(s) must have been formatted with the Disk Formatter (with system tables in specific places). As a safeguard, you may want to write–disable the unit(s) that hold your source LDU before you start. This will prevent an accidental wrong answer from overwriting the source LDU.

Start PCOPY as described earlier. If PCOPY reports an error, see the manual *AOS/VS and AOS/VS II Error and Status Messages.* The disk–to–disk dialog goes like this.

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)?

1. The *Enter today's date* prompt is only asked only by the stand–alone version of PCOPY; stand–among PCOPY gets today's date from the system calendar. If asked, type today's date, with numbers separated by one space; for example,

   11 19 93 ↵

   Will this be a disk to disk PCOPY [N] ?

2. Type Y and press NEW LINE.

   Will all disks in each LDU be on–line at all times [Y] ?

3. Press NEW LINE to accept the default value (shown in brackets).

   Specify source LDU

   Enter the name of each disk in the LDU (Press NEW LINE when done)
   Disk unit name?

4. Type the name of the first disk in the LDU; e.g., DPF0 and press NEW LINE.

   Device code [default]? (Stand–alone PCOPY only.)

   Unless you know that this disk is connected to a nondefault device code, press NEW LINE to accept the default device code. Otherwise, type the device code of the unit controller and press NEW LINE.

   Disk unit name?

   PCOPY repeats the *Disk unit name?* prompt until you respond to the prompt by pressing NEW LINE. Be sure to specify all disks in the LDU, if a multiple–disk LDU was originally created with the Disk Formatter. When you've specified all disks, press NEW LINE at the *Disk unit name?* prompt. PCOPY then describes the source LDU and asks for the destination LDU.

LDU unique ID is id

LDU name is name

Specify destination LDU.

Enter the name of each disk in the LDU (Press NEW LINE when done)

Disk unit name?

5. Type the unit name of the first disk in the destination LDU; e.g., DPF10 and press NEW LINE.

   Device code [default]? (Stand–alone PCOPY only.)

6. If the default device code shown in brackets is the correct one for this disk, press NEW LINE; otherwise, type the the proper device code and press NEW LINE.

   Disk unit name?

   As with the source LDU, PCOPY repeats the *Disk unit name?* prompt until you respond to the prompt by pressing NEW LINE. Then PCOPY displays the destination LDU's ID and name and asks for confirmation:

   LDU unique ID is id

   LDU name is name

   Copy to Disk from Disk. Please confirm (N/Y)?

7. If there are any files you want to keep on the destination LDU, type N, press NEW LINE, and begin again. PCOPY overwrites all files on the destination medium. PCOPY asks this question to give you a chance to reconsider. To proceed, type Y and press NEW LINE. When you do, PCOPY will respond

   Confirmed.

   PCOPY now copies the whole disk structure from the source to the destination LDU. A full 190–megabyte LDU takes roughly 15 minutes to copy. If PCOPY hits an error, it will display one of the error messages shown in the manual *AOS/VS and AOS/VS II Error and Status Messages.*

   NOTE: If either LDU is not ready, PCOPY will wait until you make the LDU ready.

   When PCOPY is done, it says

   Done!

8. The PCOPY is finished. If you write–disabled any disk units, you should write–enable them.

   To copy another LDU, place its disks in their units (if applicable). Then type CONTINUE and press NEW LINE (stand–alone) or XEQ PCOPY and press NEW LINE (stand–among), and return to step 1 in this section.

Example of a Disk–to–Disk PCOPY (All Disks On Line)

```
Pathname? PCOPY ↵                          (Specify PCOPY as the program
                                            that you want to run.)

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)? 11 19 93 ↵  (Stand–alone PCOPY only.)

Will this be a disk to disk PCOPY [N] ? Y ↵

Will all disks in each LDU be on–line at all times [Y]  ↵

Specify source LDU
Enter the name of each disk in the LDU (Press NEW LINE when done)
Disk unit name?  DPF0 ↵

Device code [27] ? ↵                       (Stand–alone PCOPY only.)
Disk unit name? ↵

LDU unique ID is ROOT1
LDU name is ROOT1

Specify destination LDU.
Enter the name of each disk in the LDU (Press NEW LINE when done)
Disk unit name?  DPF10 ↵
Device code [67] ? ↵                       (Stand–alone PCOPY only.)
Disk unit name? ↵

LDU unique ID is ROOT1A
LDU name is ROOT1A

Copy to Disk from Disk. Please confirm (N/Y)? Y ↵
Confirmed.

... (Time passes) ...

Done!
```

Figure 9–1 Example of a Disk–to–Disk PCOPY (All Disks On Line)

Disk–to–Disk PCOPY Dialog (All Disks Not On Line)

Use this method only if you want to copy a multiple–disk LDU, have units with removable packs, and you don't have twice as many identical disk units as there are disks in the LDU.

To copy disk to disk without all disks in each LDU on line, you need three (or more) identical disk units with removable packs. The disk with the LDU's bitmap must be in the first disk unit, and will stay in this unit throughout the LDU copy. (The bitmap disk is usually the first in the LDU, since the Disk Formatter puts the bitmap on this disk by default.) The destination disk packs must be formatted to be similar to the source disk packs (described in the section "PCOPY Requirements").

The sequence (with three disk units) goes like this:

- The bitmap disk is in the first disk unit (unit 0). You might want to write–disable this unit so that you won't accidentally overwrite it. Unit 1 (the second disk unit) holds the copy disk for the bitmap disk. Unit 2 (the third disk unit) doesn't matter for this step.
- PCOPY asks about source and destination units, and you specify unit 0 as the source and unit 1 as the destination. PCOPY copies the bitmap from the disk in unit 0 to the disk in unit 1; then prompts you to dismount unit 1.
- Remove the disk from unit 1 and put the second source disk in unit 1. The bitmap disk stays in unit 0. Write–disable disk unit 1 so that, if you accidentally specify unit 1 as the destination, the disk in unit 1 won't be overwritten. Put the second destination disk in unit 2 (the third unit). PCOPY copies the second source disk (unit 1) to the second destination disk (unit 2).
- If you're copying a two–disk LDU, you're done; remove the second disk copy from unit 2. The source LDU remains in units 0 and 1. Write–enable all write–disabled units.

If there are more disks in the LDU, remove the source disk from unit 1 and the copy disk from unit 2. Insert the next source disk in unit 1 and the next destination disk in unit 2. PCOPY copies the source in unit 1 to the destination in unit 2.

Repeat this step until you have copied all the disks in the source LDU. Then remove the last copy disk from unit 2. The bitmap source disk remains in unit 0; the last source disk remains in unit 1. Put the desired disks in units 1 and 2 to resume system operations. Write–enable all write–disabled disks.

(If you have four or more identical disk units with removable packs, you'll be able to think up other disk/unit placements for the PCOPY. This section just gives the approach. The only restriction is that the bitmap disk must remain on line and ready at all times.)

Have the LDU source and destination disk ready as described above.

Start PCOPY as described earlier. If PCOPY reports an error, see the manual *AOS/VS and AOS/VS II Error and Status Messages.* The disk–to–disk dialog goes like this.

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)?

1. The *Enter today's date* prompt appears only in the stand–alone version of PCOPY; the stand–among version of PCOPY gets today's date from the system calendar. If asked, type today's date, with numbers separated by one space; for example

   11 19 93 ↵

   Will this be a disk to disk PCOPY [N] ?

2. Type Y and press NEW LINE.

   Will all disks in each LDU be on–line at all times [Y] ?

3. Type N and press NEW LINE. (To copy disk to disk with all disks on line, see the previous section.)

   Specify source LDU
   Enter first physical disk name (Must contain bitmap):

4. Respond with the unit name that holds the first (bitmap) disk in the LDU; e.g., DPF0 and press NEW LINE.

   Device code [default] ? (Stand–alone PCOPY only.)

   Unless you know that this disk is connected to a nondefault device code, press NEW LINE for the default. Otherwise, type the device code of the unit controller. PCOPY gives the source LDU ID and name; then asks about the second LDU:

   LDU unique ID is id
   LDU name is name

   Specify destination LDU.
   Enter first physical disk name:

5. Type the unit name of the first disk in the destination LDU; for example, type DPF1 and press NEW LINE.

   Device code [default]? (Stand–alone PCOPY only.)

6. If this disk is on the default device code for its name, press NEW LINE; otherwise type the device code. PCOPY displays the destination LDU ID and name; then asks for confirmation:

   LDU unique ID is id
   LDU name is name

   Copy to Disk from Disk. Please confirm (N/Y)?

7. If there are any files you don't want overwritten on the destination disk, type N and press NEW LINE, and begin again. PCOPY overwrites all files on the destination medium. PCOPY asks this question to give you a chance to reconsider. To proceed, type Y and press NEW LINE. PCOPY will respond

   Confirmed

   PCOPY now copies the bitmap disk to the destination disk. A full 190–megabyte disk takes roughly 15 minutes to copy. If PCOPY hits an error, it will display one of the error messages explained in the manual *AOS/VS and AOS/VS II Error and Status Messages.*

   NOTE: If either LDU is not ready, PCOPY will wait until you make the LDU ready.

   When PCOPY is done, it says

   Processing completed on current disks.
   Dismount destination disk #1

   Enter source disk #2 name:

8. Dismount the destination disk (the one in the second unit, unit 1, if you followed the procedures given above). Mount the next source disk and the next destination disk. (Leave the bitmap disk alone.) Then type the source disk name; for example, DPF1 and press NEW LINE.

   Device code [default] ?

9. If this question is asked, press NEW LINE for the default code; or for a nonstandard code, type the code and press NEW LINE.

   Enter destination disk #2 name:

10. Type the name of the second destination disk; e.g., DPF2 and press NEW LINE.

    Device code [default] ?

    If this question is asked, press NEW LINE to accept the default device code shown in the brackets, or type the proper device code for this disk and then press NEW LINE.

11. PCOPY now copies the new source disk to the new destination disk. If the new source disk is the last disk in the LDU, the LDU copy is done and PCOPY responds with the message *Done!*. (PCOPY knows the number of disks in the LDU from the Disk Information Block (DIB) in the first disk.) Go to step 17.

12. If the source LDU contains more disks, PCOPY prompts you to dismount the disks and asks for the new source and destination names, as follows:

    Processing completed on current disks.
    Dismount source disk #n
    Dismount destination disk #n

    Enter source disk # n+1 Name:

13. Dismount the last source and destination disks (leaving the bitmap disk alone). Put the next source and destination disks in available units (unit 1 and 2, if you're following the procedures given above.) Then type the unit name of the next source disk; e.g., DPF1 and press NEW LINE.

    Device code [default] ?

14. If this question is asked, press NEW LINE to accept the default device code shown in brackets, or type the correct device code for this disk and press NEW LINE.

    Enter destination disk # n+1 Name:

15. Type the unit name of the next destination disk name; e.g., DPF2 and press NEW LINE.

    Device code [default] ?

16. If this question is asked, press NEW LINE for the default code or type the code. PCOPY now copies the source disk to the destination disk. If this is the last source disk in the LDU, PCOPY says *Done!*; Go to step 17. If there are more disks in the source LDU, return to step 12.

17. The PCOPY is finished. You can dismount the last copy disk and mount the disks needed for normal AOS/VS operations. Be sure to write–enable all disk units.

    To copy another LDU, ready the source and destination disk(s). Then type CONTINUE and press NEW LINE (stand–alone PCOPY) or type XEQ PCOPY and press NEW LINE (stand–among PCOPY) and return to step 1 in this section.

Example of a Disk–to–Disk PCOPY (All Disks Not On Line)

This example shows a PCOPY of a two–disk LDU.

Pathname? PCOPY ↵

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)? 11 19 93 ↵ (Stand–alone PCOPY only.)
Will this be a disk to disk PCOPY [N] ? Y ↵
Will all disks in each LDU be on–line at all times [Y] N ↵

Specify source LDU
Enter first physical disk name (Must contain bitmap): DPF0 ↵
Device code [27] ? (Stand–alone PCOPY only.)

LDU unique ID is SYS
LDU name is SYS

Specify destination LDU.
Enter first physical disk name: DPF1 ↵
Device code [27] ? ↵ (Stand–alone PCOPY only.)

LDU unique ID is SYS1
LDU name is SYS1

Copy to Disk from Disk. Please confirm (N/Y)? Y ↵
Confirmed.

... (Time passes) ...

Processing completed on current disks.
Dismount destination disk # 1

Dismount the destination disk; mount the new source and destination disks.

Enter source disk #2 name: DPF1 ↵
Device code [27] ? ↵ (Stand–alone PCOPY only.)
Enter destination disk # 2 name: DPF10 ↵
Device code [67] ? ↵ (Stand–alone PCOPY only.)

... (Time passes) ...

Done!

Figure 9–2 Example of a Disk–to–Disk PCOPY (All Disks Not On Line)

Disk–to–Tape PCOPY Dialog

Have all the tape units set to DENSITY HIGH (if this applies), to conserve tape. Then mount all the tapes you can and put them on line. Mounting multiple tapes reduces the number of tape changes you must make. Next, start PCOPY as described above. The disk–to–tape PCOPY dialog goes as follows.

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)?

1. This date question is asked only by stand–alone PCOPY; stand–among PCOPY gets today's date from the system calendar. If asked, type today's date, with numbers separated by one space; for example

   11 19 93 ↵

   Will this be a disk to disk PCOPY [N]?

2. Press NEW LINE to accept the default answer, No, shown in square brackets.

   Is source a logical Disk(D), labeled Tape (T), or Floppy(F)?

3. Type D (for logical disk) and press NEW LINE.

   Specify source LDU
   Enter the name of each disk in the LDU (Press NEW LINE when done)
   Disk unit name?

4. Type the name of the first disk in the LDU; e.g., DPF0 and press NEW LINE.

   Device code [default]?

5. This question is skipped by stand–among PCOPY. Unless you know that this disk is connected to a nonstandard device code, press NEW LINE for the default. Otherwise, type the device code of the unit controller.

   Disk unit name?

   PCOPY repeats the *Disk unit name?* prompt until you respond to the prompt by pressing NEW LINE. Be sure to specify all disks in the LDU, if a multiple–disk LDU was originally created with the Disk Formatter.

   When you've specified each disk, press NEW LINE at the *Disk unit name?* prompt. PCOPY then describes the source LDU and asks for the destination medium:

   LDU unique ID is id
   LDU NAME is name

   Copy to Disk (D), labeled Tape (T), or Floppy (F)?(Stand–alone PCOPY only.)

   Type T (for labeled tape) and press NEW LINE.

6. *Tape unit name?*

   You must specify a tape unit name and volume ID for each tape PCOPY will use. This means that if you'll need 10 tapes, you'll need to specify 10 tape unit names and 10 volume IDs.

   If you have several tape units, you can specify their unit names and the volume IDs in sequence. PCOPY will then copy to them in sequence, allowing you to leave the terminal for a while. For example, you might type the sequence MTB0, PCOP01, MTB1, PCOP02, MTB2, and PCOP03, pressing NEW LINE after each.

   Then PCOPY would copy to all tapes needed. If PCOPY needed more tapes, it would ask *Tape unit name?* again. Then, you'd remove the tapes, mount the second set of tapes, and type MTB0, PCOP04, MTB1, PCOP05, MTB2, and PCOP06, pressing NEW LINE after each.

   After PCOPY has filled the first tape or tape set, the *Tape unit name?* prompt indicates that it's time to dismount the current tape set and mount the next one. When you see this prompt, be sure to dismount the newly filled tape or tape set, and mount the new one, before continuing. If you don't do this, you might accidentally type a unit name and volume ID, and PCOPY would write new material to the tape. This would invalidate the whole backup and you'd need to restart it.

   When you type the unit name, omit @; for example,

   MTC0 ↵

   Device code ? [22] (Stand–alone PCOPY only.)

   For a tape unit on the default device code, press NEW LINE. For a tape unit on any other device code, type the device code and press NEW LINE.

7. For an MTD unit, PCOPY asks

   Recording density: 0=6250, 1=1600, 2=800 [0] ?

   Generally, with an MTD unit, you should select the highest density (6250 b/in); if so, press NEW LINE. For a lower density (which will consume much more tape), type the appropriate number and press NEW LINE. PCOPY doesn't ask this question again for this copy.

   Specify volume ID?

8. You must specify a volume ID for each tape reel or cartridge. Since you usually need several volumes of tape for a PCOPY, a naming convention — like PCOP01, PCOP02, PCOP03, and so on — is desirable.

   Type the volume ID that you want on the tape. When PCOPY labels the tapes, it will write this volume ID to the tape.

   Tape volume IDs are one to six filename characters long. The volume ID should be written somewhere on a paper label on the tape reel or cover; if it's not there, write it there now.)

For example, for the first volume, you might type PCOP01 and press NEW LINE.

Tape unit name?

9. PCOPY repeats the *Specify volume ID* and *Tape unit name?* prompts until you respond to the *Tape unit name?* prompt by pressing NEW LINE. Press NEW LINE unless you have another unit and volume to identify (as described above).

Expiration date (default is mm dd yy)

10. The default expiration date is 90 days from the current date. If you try to copy to a tape whose expiration date has not been reached, PCOPY will ask for confirmation before proceeding.

    The expiration date can be a useful tool — a time limit that can be overridden. You can select the 90–day default by pressing NEW LINE. Or you can type the date after which you want the tape to be reused; e.g., 12 31 93 and press NEW LINE. Or, you can type 0 and press NEW LINE, which marks the tape as already expired, so it can be reused at any time. If you specify a date (not 0), it must be at least one day later than today's date. After you specify an expiration date, PCOPY asks

    Copy to Tape from Disk. Please confirm (N/Y)?

11. PCOPY gives you this chance to confirm the tape write. To confirm, type Y and press NEW LINE. PCOPY will reply

    Confirmed.

    What buffer size do you want in Kbytes (1, 2, 4, 8, 16 or 32) [n]

12. This question sets the buffer size for the copy. A larger buffer size means a larger record, which reduces the number of records; this saves tape by reducing the number of interrecord gaps.

    You can choose the maximum buffer size selected at VSGEN, but doing so may limit your choices. For example, for MTD units, the VSGEN default maximum buffer size is 32 Kbytes — which produces a backup you can load only with stand–among PCOPY. If you choose 16 Kbytes, you can use standamong or standalone PCOPY to reload your backup.

    The recommended sizes are

Unit Type	VSGEN Default Buffer Limit	Recommended Buffer Size
MTD	32 Kbytes	16 Kbytes
MTB	8 Kbytes	16 Kbytes
MTC	8 Kbytes	16 Kbytes
MTJ	16 Kbytes	16 Kbytes

    If you need to restore from this PCOPY tape set, PCOPY will select the correct buffer size automatically when you restore.

For all units, type 16 and press NEW LINE.

Do you want data compression (Y/N)?

13. For MTJ tape units that support data compression, PCOPY asks if you want to use data compression. Hardware data compression is advisable on these drives, so press NEW LINE unless you want to transport data to a drive that cannot read compressed data.

What density do you want (0=No change, 1=Low, 2=Medium, 3=High)? [3]

14. For MTJ tape units that support multiple densities, PCOPY asks if you want to set the tape density. Generally, select High by pressing NEW LINE.

Generation number [current]

15. The generation number is the part of the tape header label that describes the number of copies to the tape. It is an indicator of tape "mileage." PCOPY resets the generation number to 0001 automatically.

    To have PCOPY use the displayed number, press NEW LINE. To specify a different number, type the number and press NEW LINE.

    After you reply to the *Generation number* prompt, PCOPY starts copying the LDU to labeled tape. A 2400–foot tape takes 6 to 9 minutes; an 800–foot tape on an MTC unit takes 5 to 6 minutes. If PCOPY hits an error, it will display one of the error messages shown in the manual *AOS/VS and AOS/VS II Error and Status Messages.*.

    NOTE: If either LDU is not ready, PCOPY will wait until you make the LDU ready.

16. After PCOPY fills each tape, it rewinds the tape; then asks for the next unit and volume ID (unless you specified a set as shown in step 6 above). You'll mount the next tape volume(s), specify the unit names(s) and volume ID(s), and PCOPY will proceed.

    When PCOPY has copied the entire LDU to tape, it says

    PCOPY to tape complete. Do you want to verify the tape(s)? [N]

17. PCOPY can verify the tapes that it wrote by reading them back. If all tapes read back without errors, you can be nearly certain that they will restore properly later. (Bad storage conditions can make tapes deteriorate after you've verified them.) On most tapes, verifying takes about as long as copying, and is especially recommended for model 6351 tape drives. (Verifying is faster on ultra high–capacity drives.)

When you verify, PCOPY must read the tapes in the same order that it wrote them. So, for a multiple–tape copy, you must dismount the last tape(s) and mount the first tape(s). PCOPY then reads the label and tape, and prompts for others as needed. When done, it will confirm the verification and proceed to step 18. (If PCOPY hits a fatal read error, replace the bad tape with another, restart PCOPY if needed, and redo the PCOPY dump from the beginning, at step 6.)

If you decide to verify the tapes, type Y and press NEW LINE, and then go to step 6. If you don't want to verify the tapes, press NEW LINE or type N and press NEW LINE.

18. When PCOPY is finished, it displays the message

Done!

Remove the write–enable rings from the tapes (if this applies); make sure they have paper labels with the correct volume IDs; and store them in a safe place.

To copy another LDU, place its disks in their units (if applicable). Then type CONTINUE and press NEW LINE (stand–alone) or XEQ PCOPY and press NEW LINE (stand–among) and return to step 1 in this section.

Example of a Disk–to–Tape PCOPY

This example shows how to perform a disk–to–tape PCOPY.

Pathname? PCOPY ↵

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)? 11 19 93 ↵ (Stand–alone PCOPY only.)
Will this be a disk to disk PCOPY [N] ? ↵
Is source a logical Disk (D), labeled Tape (T), or Floppy (F)? D ↵

Specify source LDU
Enter the name of each disk in the LDU (Press NEW LINE when done)
Disk unit name? DPJ0 ↵
Device code ? [24] ↵ (Stand–alone PCOPY only.)
Disk unit name? ↵

LDU unique ID is 1
LDU name is ROOT

Copy to Disk (D), labeled Tape (T), or Floppy (F) ? T ↵
Tape unit name? MTC0 ↵
Device code? [22] ↵ (Stand–alone PCOPY only.)
Specify Volume ID? PCOP01 ↵
Tape unit name? ↵
Expiration date (default is 2 17 94) ↵

Copy to Tape from Disk. Please confirm (N/Y)? Y ↵
Confirmed.

What buffer size do you want in Kbytes (1, 2, 4, 8) [8] 2 ↵

Generation number? [0003] ↵

... (Time passes as PCOPY writes to tape) ...

Tape unit name?

Remove the first tape, and mount the next tape. Type

MTC0 ↵

Device code? [22] ↵
Specify volume ID ? PCOP02 ↵ (Second tape volume ID.)

... (PCOPY writes to second tape) ...

PCOPY to tape complete. Do you want to verify the tape(s)? [N]

Done!

Figure 9–3 Example of a Disk–to–Tape PCOPY

Tape–to–Disk PCOPY Dialog

Mount all the labeled tapes that you can and put them on line. Mounting multiple tapes will reduce the number of tape changes you must make.

As described earlier, in the section "PCOPY Requirements," the destination LDU must have been formatted with the Disk Formatter, with system tables in specific places. If not, you may need to run the Formatter to move move one or more tables. PCOPY will tell you what to move.

If your system LDU has AOS/VS files on it, try to start PCOPY from disk (by executing the BOOT sequence for stand–alone; using XEQ PCOPY for stand–among). Stand–among PCOPY can restore only a nonmaster LDU.

If your system LDU doesn't have AOS/VS files on it (for example, you're rebuilding it), you must start PCOPY from the PCOPY tape you made earlier. Do it as described early in this chapter, in the section "Starting PCOPY from Tape." Then return here and proceed as follows.

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)?

1. The *Enter today's date* prompt appears only in the stand–alone version of PCOPY; the stand–among version of PCOPY gets today's date from the system calendar. If asked, type today's date, with numbers separated by one space; for example,

   11 19 93 ↵

   Will this be a disk to disk PCOPY [N]?

2. Press NEW LINE to select the default answer, No, shown in square brackets.

   Is source a logical Disk(D), labeled Tape (T), or Floppy (F)?

3. Type T (labeled tape) and press NEW LINE.

   Tape unit name?

4. Type the name of the tape unit that holds the first tape; e.g., MTC0, and press NEW LINE.

   MTC0 ↵

   Device code ? [22] (Stand–alone PCOPY only.)

   For a tape unit on the default device code, press NEW LINE. For a tape unit on any other device code, type the device code and and press NEW LINE.

   Specify volume ID?

5. Type the volume ID that's on the tape. This should be visible on the paper label. (If you can't determine the volume ID, you can make one up; then, when PCOPY reads the label, the error message will tell you the actual volume ID.) But remember that, to restore the LDU, the tapes must be copied to the LDU in the same order that they were copied from the LDU. When PCOPY copies an LDU to tape, it keeps the LDU ID as the fileset identifier. This lets PCOPY avoid restoring the wrong material to an LDU.

   For example, if the volume ID were PCOP01, you'd type PCOP01 and press NEW LINE.

   Tape unit name?

6. PCOPY repeats the *Specify volume ID?* and *Tape unit name?* prompts until you respond to the *Tape unit name?* prompt by pressing NEW LINE.

   If you have several tape units, you can specify their unit names and the volume IDs in sequence; PCOPY will then copy to them in sequence, allowing you to leave the terminal for a while. For example, you might type the sequence MTB0, PCOP01, MTB1, PCOP02, MTB2, and PCOP03, pressing NEW LINE after each.

   Then PCOPY would copy to all tapes needed. If more tapes were needed to restore the LDU, PCOPY would display the *Tape unit name?* prompt again. Then, you'd remove the tapes, mount the second set of tapes, and type MTB0, PCOP04, MTB1, PCOP05, MTB2, and PCOP06, pressing NEW LINE after each.

   When you've specified each unit and volume mounted, press NEW LINE in response to the *Tape unit name?* prompt. PCOPY will then display

   Specify destination LDU
   Enter the name of each disk in the LDU (Press NEW LINE when done)
   Disk unit name?

7. Type the unit name of the first disk in the LDU; e.g., DPF0 and press NEW LINE.

   Device code [default] ? (Stand–alone PCOPY only.)

8. Unless you know that this disk is connected to a nondefault device code, press NEW LINE to accept the default. Otherwise, type the device code of the unit's controller, and then press NEW LINE.

   Disk unit name?

9. PCOPY repeats the *Disk unit name?* prompt until you respond by pressing NEW LINE to the *Disk unit name?* prompt. Be sure to specify all disks in the LDU, if a multiple–disk LDU was originally created with the Disk Formatter.

When you've specified each disk in the LDU, press NEW LINE at *Disk unit name?* prompt. PCOPY displays the LDU unique ID and name, then asks for confirmation.

LDU unique ID is id
LDU name is name

Copy to Disk from Tape. Please confirm (N/Y)?

10. If there are any files you want on the destination LDU, type N (and press NEW LINE), and begin again. PCOPY overwrites all files on the destination medium. PCOPY asks this question to give you a chance to reconsider. To proceed, type Y and press NEW LINE. PCOPY will respond with

    Confirmed

    PCOPY now starts copying the tape(s) to the LDU. A 2400–foot tape takes 7 to 10 minutes. An 800–foot tape on an MTC unit tape takes 5 to 6 minutes. If PCOPY hits an error, it will display one of the error messages explained in the *AOS/VS and AOS/VS II Error and Status Messages* manual.

    NOTE: If either LDU is not ready, PCOPY will wait until you make the LDU ready.

    As PCOPY reaches the end of each tape, it will rewind the tape. Then, if it needs more tape volumes, it will return to the *Tape unit name?* prompt as shown in step 3 above.

    When PCOPY is finished, it displays the message

    Done!

11. To restore another LDU, place its disks in their units (if applicable). Then type CONTINUE and press NEW LINE (stand–alone) or XEQ PCOPY and press NEW LINE (stand–among) and return to step 1 in this section.

Example of a Tape–to–Disk PCOPY

This is an example of to do a tape–to–disk PCOPY.

Pathname? PCOPY ↵

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)? 11 19 93 ↵ (Stand–alone PCOPY only.)

Will this be a disk to disk PCOPY [N]? ↵
Is source a logical Disk (D), labeled Tape (T), or Floppy (F)? T ↵
Tape unit name? MTC0 ↵
Device code ? [22] ↵
Specify volume ID? PCOP01 ↵ (Type tape volume ID.)
Tape unit name? ↵
Specify volume ID? PCOP02 ↵ (Type tape volume ID.)

Specify destination LDU
Enter the name of each disk in the LDU (Press NEW LINE when done)
Disk unit name? DPJ0 ↵
Device code [24] ? ↵ (Stand–alone PCOPY only.)
Disk unit name?

LDU unique ID is UDD1
LDU name is UDD1

Copy to Disk from Tape. Please confirm (N/Y)? Y ↵
Confirmed.

... (time passes as PCOPY copies to the LDU) ...

Tape unit name?

Dismount tape from unit 0; mount the next tape on unit 0.

MTC0 ↵

Device code ? [22] ↵
Tape unit name? ↵

... (Time passes as PCOPY copies to the LDU) ...

Done!

Figure 9–4 Example of a Tape–to–Disk PCOPY

Disk–to–Diskette PCOPY Dialog

Before starting, you need enough hardware–formatted diskettes to hold your LDU's files. These diskettes should also have been formatted with the Disk Formatter. Running a full format, with at least one pattern, on a diskette allows PCOPY to use a diskette that has bad blocks. If the Formatter hasn't run on a diskette, and PCOPY discovers a bad block on it, PCOPY will abort and you must restart the PCOPY operation from the beginning. When you run the Disk Formatter on a diskette destined for PCOPY, answer N and press NEW LINE to the Formatter's *System Disk* question.

For a full 39–megabyte Winchester disk, you need about 40 737,000–byte diskettes. You should write the number of each diskette on its paper label — for example, 1, 2, 3, 4, and so on. When you write on a label that's already attached to the diskette inner envelope, use a felt–tipped pen to avoid scoring the diskette surface. Numbering is essential; if you ever need to restore an LDU from these diskettes, PCOPY will require the diskettes in the original dump sequence.

NOTE: In order to use PCOPY with diskettes, you must be running the 16–bit CLI under AOS/VS. The 32–bit CLI does not support diskettes.

When you're ready, start PCOPY as described in the section "Starting PCOPY from Disk."

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)?

1. The *Enter today's date* prompt is asked only by stand–alone PCOPY; stand–among PCOPY gets today's date from the system calendar. If asked, type today's date, with numbers separated by one space — for example,

   11 19 93 ↵

   Will this be a disk to disk PCOPY [N]?

2. Press NEW LINE for the default answer, No, shown in square brackets.

   Is source a logical Disk (D), labeled Tape (T), or Floppy(F)?

3. Type D (for logical disk) and press NEW LINE.

   Specify source LDU
   Enter the name of each disk in the LDU (Press NEW LINE when done)
   Disk unit name?

4. Type the name of the first disk in the LDU. This is DPJ0 (system disk) or DPJ1 (for the second disk, if any). For example,

   DPJ0 ↵

   Device code? [24]

5. This question is skipped by stand–among PCOPY. If asked, press NEW LINE to accept the default device code.

Disk unit name?

6. PCOPY repeats *the Disk unit name?* prompt until you respond to the prompt by pressing NEW LINE.

   Usually, there's only one disk in the LDU. If so, press NEW LINE now. PCOPY then describes the LDU's ID and name, and asks for the destination medium:

   Copy to Disk (D), labeled Tape (T) or Floppy (F)?

7. Type F (for floppy disk) and press NEW LINE.

   Floppy unit name? [DPJ10]

   The primary diskette unit name is DPJ10; the secondary unit name is DPJ11. Accept the default by pressing NEW LINE.

   Device code? [64]

8. This question is skipped by stand–among PCOPY. If asked, press NEW LINE to accept the default device code.

   Specify volume ID

9. Type the name of the volume ID that you want on the first diskette. This volume ID will identify the entire diskette copy. It must be between one and six characters long. You might use the date (perhaps in form yy–mm–dd) or any other name that will help identify the copy. For example, for November 19, 1993, you might type the volume ID 931119 and press NEW LINE.

   The volume ID will be required to restore the LDU, so make sure it's written on the paper label of diskette number 1.

   Copy to Floppy from Disk. Please confirm (N/Y)?

10. PCOPY gives you this chance to confirm the write to diskette. To confirm, type Y and press NEW LINE. PCOPY will respond

    Confirmed.

    PCOPY now starts copying the LDU to diskette. Each diskette takes about 2 minutes. If PCOPY hits an error, it will display one of the error messages shown later, in the *AOS/VS and AOS/VS II Error and Status Messages* manual.

    NOTE: If either LDU is not ready, PCOPY will wait until you make the LDU ready.

11. After PCOPY has filled the diskette, it says

    Mount next floppy, strike any key when ready.

    Remove the diskette from its unit and write the diskette sequence number on the diskette envelope if you have not done so already (use a felt–tipped pen). Return the diskette to an outer envelope. Insert the next diskette in the unit.

12. Press any key to continue.

    PCOPY now copies from the LDU to the new diskette. When done, it displays the message shown in step 11. Repeat steps 11 and 12 until PCOPY says

    Done!

    Remove the last diskette from its unit, and store all diskettes safely, away from strong magnetic fields.

    To copy another LDU, type CONTINUE and press NEW LINE (stand–alone PCOPY) or XEQ PCOPY and press NEW LINE (stand–among PCOPY). Make sure you have enough diskettes, and return to step 1 in this section.

Example of a Disk–to–Diskette PCOPY

Pathname? PCOPY ↵

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)? 11 19 93 ↵ (Stand–alone PCOPY only.)

Will this be a disk to disk PCOPY [N] ? ↵
Is source a logical Disk (D) labeled Tape (T), or Floppy(F)? D ↵

Specify source LDU
Enter the name of each disk in the LDU (Press NEW LINE when done)
Disk unit name? DPJ0 ↵
Device code? [24] ↵ (Stand–alone PCOPY only.)
Disk unit name? ↵

LDU unique ID is 1
LDU name is ROOT

Copy to Disk (D), labeled Tape (T), or Floppy (F)? F ↵
Floppy unit name? [DPJ10] ↵
Device code? [64] ↵ (Stand–alone PCOPY only.)
Specify volume ID? 931119 ↵

Copy to Floppy from Disk. Please confirm (N/Y)? Y ↵
Confirmed.

... (Time passes) ...

Mount next floppy, press any key when ready.

Mount next diskette and press key.

... (Time passes — you insert diskettes) ...

Done!

Figure 9–5 Example of a Disk–to–Diskette PCOPY

Diskette–to–Disk PCOPY Dialog

Before you start, get the set of diskettes that PCOPY copied to. The volume ID used for the dump should be visible on the first diskette's paper label, and the sequence number of each subsequent diskette should be visible on its paper label.

As described earlier, in the section "PCOPY Requirements", the destination LDU must have been formatted with the Disk Formatter, with system tables in specific places. If not, you may need to run the Formatter to move one or more tables. PCOPY will tell you what to move.

If your system LDU has AOS/VS files on it, try to start PCOPY from disk (BOOT 24 and press NEW LINE for stand–alone; XEQ PCOPY and press NEW LINE for stand–among). Stand–among PCOPY can restore only a nonmaster LDU; for example, unit @DPJ1.

If your system LDU doesn't have AOS/VS files on it (for example, you're restoring it), you must start PCOPY from the PCOPY diskette that you made earlier. Follow the steps described in the section "Starting PCOPY from Diskette." Then return here and proceed as follows.

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)?

1. The *Enter today's date* prompt appears only in the stand–alone PCOPY; stand–among PCOPY gets today's date from the system calendar. If asked, type today's date, with numbers separated by one space; for example

   11 19 93 ↵

   Will this be a disk to disk PCOPY [N]?

2. Press NEW LINE for the default answer, No, shown in square brackets.

   Is source a logical Disk (D), labeled Tape (T), or Floppy (F)?

3. Type F (for floppy disk) and press NEW LINE.

   Floppy unit name [DPJ10]?

   The primary diskette unit name is DPJ10; the secondary unit name is DPJ11. Press NEW LINE to accept the default floppy name, or type the correct diskette unit name.

   Device code ? [64]

4. This appears in the stand–alone version of PCOPY only. If asked, press NEW LINE to accept the default device code.

   Specify volume ID?

5. Type the volume ID that was specified for the diskette dump. This should be written on the paper label of the first diskette. (If you can't determine the volume ID, make one up; later, when PCOPY checks the diskette, it will display the real ID in an error message. You can then use CTRL–C CTRL–A to interrupt PCOPY, restart PCOPY and type the actual volume ID. But remember that, to restore an LDU properly, the diskettes must be copied to the LDU in the same order that they were copied from the original LDU.)

   For example, if the label were 931119, you'd type

   931119 ↵

   Specify destination LDU
   Enter the name of each disk in the LDU (Press NEW LINE when done)
   Disk unit name?

5. Type the name of the first disk in the LDU. This is DPJ0 (system disk) or DPJ1 (for the second disk, if any). For example,

   DPJ0 ↵

   Device code? [24]

6. This is skipped by stand–among PCOPY. If asked, press NEW LINE to accept the default device code.

   Disk unit name?

7. PCOPY repeats the *Disk unit name?* prompt until your respond to the prompt by pressing NEW LINE. Usually, there will be only one disk in the LDU; if so, press NEW LINE.

   Copy to Disk from Floppy. Please confirm (N/Y)?

8. If there are any files you want on the destination LDU, type N and press NEW LINE. PCOPY overwrites all material on the destination medium. PCOPY gives you this chance to confirm the write to disk. To confirm, type Y and press NEW LINE. PCOPY will respond with

   Confirmed.

   PCOPY now checks the volume ID and sequence number. If these are correct, it starts copying the diskette to the LDU. If PCOPY hits an error, it will display one of the error messages shown in the manual *AOS/VS and AOS/VS II Error and Status Messages.*

   NOTE: If either LDU is not ready, PCOPY will wait until you make the LDU ready.

9. After PCOPY has finished copying from the diskette, it says

   Mount next floppy, press any key when ready.

   Remove the diskette from its unit and replace it in the envelope. Insert the next diskette in the unit.

10. Press any key to continue.

    If the sequence number is correct, PCOPY now copies from the new diskette to the LDU. (If the sequence number is wrong, PCOPY will display an error message and prompt for the correct diskette. Try to find and mount the correct diskette. If you succeed, fine; if not, you will have to restart the PCOPY from the first diskette).

    When PCOPY has finished copying from this diskette, it displays the *Mount next floppy* prompt shown in step 9. Repeat steps 9 and 10 until until PCOPY says

    Done!

    Remove the last diskette from its unit and store all diskettes safely.

11. To restore another LDU, type CONTINUE and press NEW LINE or XEQ PCOPY and press NEW LINE (stand–among). Return to step 1 in this section.

Example of a Diskette–to–Disk PCOPY

This is an example of how you can use PCOPY to copy data between a diskette and a hard disk.

Pathname? PCOPY ↵

AOS/VS Disk Copier REV n

Enter today's date (mm dd yy)? 11 19 93 ↵ (Stand–alone PCOPY only.)
Will this be a disk to disk PCOPY [N] ?
Is source a logical Disk (D) labeled Tape (T), or Floppy(F)? F ↵

Floppy unit name? [DPJ10] ↵
Device code ? [64] ↵ (Stand–alone PCOPY only.)
Specify volume ID? 931119 ↵
Specify destination LDU
Enter the name of each disk in the LDU (Press NEW LINE when done)
Disk unit name? DPJ0 ↵
Device code? [24] ↵ (Stand–alone PCOPY only.)
Disk unit name? ↵

LDU unique ID is 1
LDU name is ROOT

Copy to Disk from Floppy. Please confirm (N/Y)? Y ↵
Confirmed.

... (Time passes) ...

Mount next floppy, press any key when ready.

Mount next diskette; strike key.

... (Time passes) ...

Done!

Figure 9–6 Example of a Diskette–to–Disk PCOPY

End of Chapter

Chapter 10
Improving System Availability

Read this chapter

- When you want to understand the concept of "improved system availability";
- When you want to learn how LDUs can be shared by two computer systems;
- When you want to learn how to use multiported disk units or disk mirroring;
- When you want to learn about new features of the operating system that improve system availability.

AOS/VS and AOS/VS II offer a number of methods for reducing a single point of failure. Your choice of any of these methods involves an understanding of what the choices and costs are. This chapter provides a background in high availability.

Major sections in this chapter are

- Implementing High Availability
- Physical Disks and LDUs
- Using Multiported Disks
- Using Mirrored LDUs
- Using Backup Disk Controllers and Mirroring (AOS/VS II Only)
- Using MRC Configurations (AOS/VS II Only)
- Using CLARiiON™ or H.A.D.A./MV Configurations
- Forcing the Release of an LDU (AOS/VS II Only)
- Using the Runtime Configuration Manager (AOS/VS II and MRC Only)
- Using Automatic Dump and Automatic Reboot (AOS/VS II Only)
- Automatic IAC Reboot

Implementing High Availability

The term availability as used in this chapter means "usability of the system" or "the system's availability for processing." Most measures of availability are based on MTBF (mean time between failures), where availability equals MTBF minus system downtime. Data General defines "availability" in terms of the operating system, not application software. If the operating system is running, DG would consider the system to be "available." If the operating system fails, it should be rebootable within 15 minutes. For traditional minicomputer systems, the mean–time–between–failures is measured in months.

Data General's high availability goals are to

- Prevent failure of a single component (like a job processor or disk controller) from disabling the entire system. We do this by offering computer systems with multiple job processors and allowing configuration of redundant components (for example, two disk controllers multiported to the same disk unit);
- Provide recovery within 15 minutes, if failure occurs;
- Allow repair without disrupting processing if hardware failure occurs.

Obtaining High Availability

Availability involves hardware, operating system software, and application software. Hardware is critically important. To obtain high availability from hardware, you must have appropriate backup hardware installed and configured in your computer system. The Data General hardware that provides the highest availability are such high–end computer systems as the ECLIPSE MV/60000™ HA, the ECLIPSE MV/40000™ HA, the MRC for I/O, a CLARiiON™ or H.A.D.A./MV high–availability disk array, and a CLARiiON tape array storage system. Whatever your hardware configuration, you can improve system availability by using techniques like multiporting (explained later in this chapter).

To obtain high availability from operating system software, you must configure the operating system to use its own high–availability features like automatic reboot (which you specify during VSGEN), disk mirroring, and multiporting disks (which requires installing secondary disk controllers).

To obtain high availability from application software that you develop, your programmers must code for it (that is outside the scope of this manual). Data General's application software (like INFOS II, CEO, and XTS) varies in its ability to withstand and recover from failures.

If high availability is critically important to you, you can improve the availability of the operating system by

- Acquiring a computer with high availability features like multiple job processors and integrated diagnostic capabilities;
- Acquiring multiple I/O systems like disk controllers and disk drives; and,
- Using disk mirroring and/or secondary disk controllers with multiported disk units.

Acquiring hardware is beyond the scope of this manual. However, the remaining sections of this chapter explain features of the operating system that help you improve system availability.

Physical Disks and Logical Disk Units (LDUs)

The AOS/VS and AOS/VS II file systems are built using LDUs that reside on physical disks. A physical disk is the magnetic disk contained in your system's disk drive(s), whether it be platters, cartridge, or diskette. An LDU is one or more physical disks (or pieces of one or more physical disks), that are logically linked together as a unit.

For any LDU, all physical disks that make up an LDU must be on line and initialized before you can access the LDU.

With AOS/VS you can only create LDUs that are equal in size to one, two, or more physical disks. You cannot create an LDU smaller than a physical disk.

With AOS/VS II, LDUs need not encompass entire physical disks. The AOS/VS II file system allows for LDUs smaller than, equal to, or larger than a single physical disk. These LDUs do not have to be contiguous; they can be composed of separate pieces, up to a maximum of eight, which can reside anywhere on any of the physical disks.

Single–Disk and Multiple–Disk LDUs

You already know (if you've read the chapter in the *Installing* manual for your particular operating system that describes the Disk Formatter (AOS/VS) or the Disk Jockey (AOS/VS II) utility) that you can create a single–disk or multiple–disk LDU. A multiple–disk LDU can include as many as eight physical disks.

The disk drive(s) you use to format a disk is irrelevant to the LDU. For example, with removable disk packs, you can format an LDU in the drive designated DPF0 and run it in DPF11; or you can format a two–disk LDU in drives DPF12 and DPF23 and run it in DPF1 and DPF2. During system startup, you are asked to specify each additional disk in the LDU by name and device code. When you initialize an LDU from the CLI, you specify the disk unit name. Configuring and managing your system is much simpler if you keep each physical disk in an LDU in the same disk unit all the time, but there is no requirement that prevents you from swapping a disk between different disk units. You should keep the system LDU in disk unit 0, however.

The real advantage of a multiple–disk LDU is that it allows a contiguous file to span more than one physical disk. Some Data General data management products such as INFOS II and DG/DBMS may need such huge contiguous files. If your site will use such a file, you will need to build a multiple–disk LDU for it. You would run this multiple–disk data LDU in addition to a single–disk system LDU.

The following sections explain two valuable aspects of LDUs: how to share an LDU between two different computer systems, and how to mirror an LDU. Both of these techniques are designed to give you higher data availability.

Using Multiported Disks

It is possible to share one or more LDUs composed of certain model disks between two ECLIPSE MV/Family CPUs. (Each computer must have a system disk of its own.) The main advantage of this dual–port arrangement is fast recovery if one system fails.

CAUTION: Do not multiport system disks because multiport protection is not available when booting.

Each computer system runs independently, but you need to build the system disks, the configuration files, and run VSGEN so that each system that is going to share the LDU is aware of the other system. The *Installing* manual for your particular operating system explains how to create, format, and name multiported disks, and generate a system that uses them.

You can format the multiported disk from either computer, and via the Disk Formatter (AOS/VS) or the Disk Jockey (AOS/VS II) utility, giving the LDU a unique ID and meaningful LDU filename. Give the multiported disk a restrictive ACL (for example, null). This ACL will help prevent unauthorized users from initializing the LDU.

You also must protect the LDUs' :PER entries. It's possible, before initialization, for a user to read or write to any of the LDU's disks as a physical device (for example, by typing DUMP_II/V @DPJ2 MYFILE and pressing NEW LINE). To prevent this, the UP macro in both computer systems should set the unit ACL(s) in :PER to null or for very specific access. For example,

```
ACL/K @DPJ2
```

When one system wants to use the LDU, it turns on the Superuser privilege, initializes the LDU, and changes its ACL as needed. Then users and programs can access it like any LDU, by directory name. After initialization, the LDU is part of the file system, and access controls remain in force.

An LDU cannot be initialized if it's already initialized — this prevents the second system from initializing the multiported disk while the first system is using it. After a system has released the multiported disk, it can be initialized by either system.

Example of a Multiported Disk Application

Assume that two MV/Family systems, each with two 354–Mbyte LDUs, share a 354–Mbyte LDU as shown in Figure 10–1.

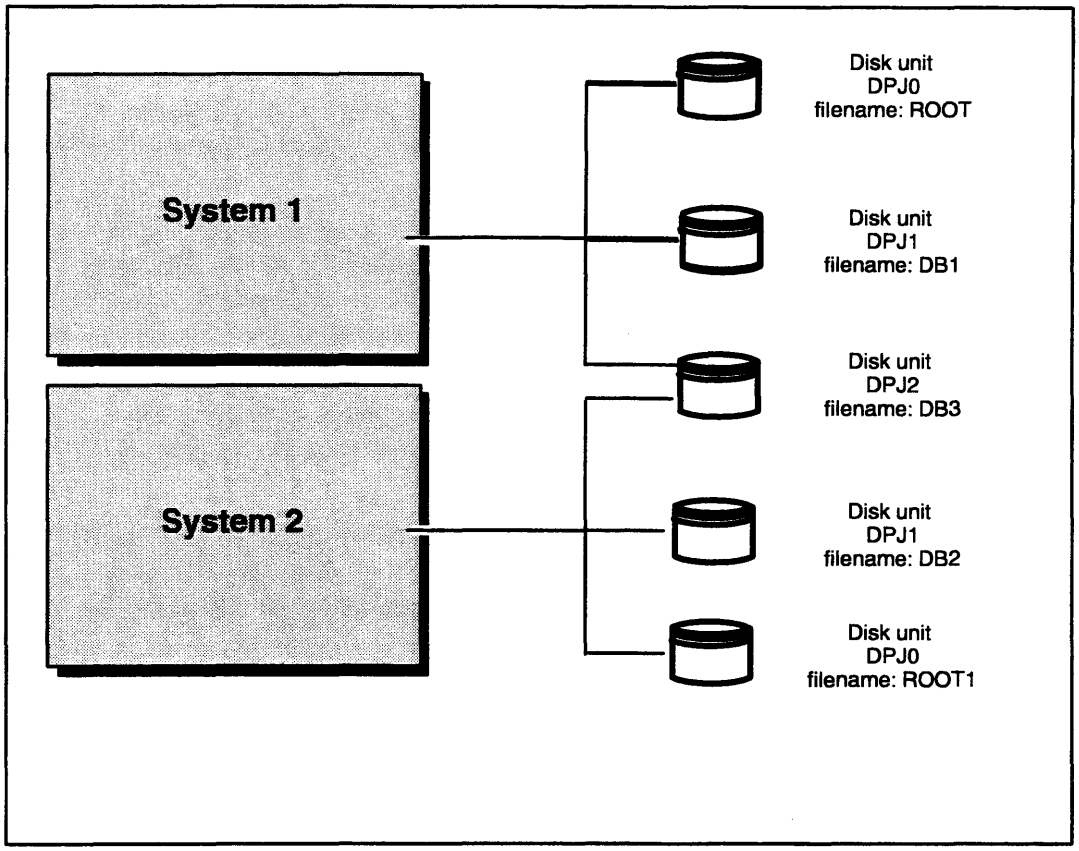


Figure 10–1 Two ECLIPSE MV/Family Systems with a Multiported Disk

The site uses the two systems and shares an LDU as follows:

8:30 a.m. – 5.00 p.m.	During a normal working day, System 1 is used for interactive access and update of LDU DB1 (the primary database). Three times each day, during coffee breaks and lunch, System 1 initializes the multiported disk, copies the updated database to it, and releases it.
	System 2 is used for program development, with a copy of the most current database on its database LDU, DB2. System 2 has a copy of System 1's applications programs.
5:00 p.m.	As the working day ends, the load slackens on System 1. Once again, System 1 copies the DB1 database to the multiported disk and releases it. System 1 will remain up through the second and third shifts.
5:30 p.m.	System 2 initializes the multiported disk and copies the DB3 database to its database LDU DB2. Then, System 2 backs up the multiported disk using its own tape unit(s).
1:00 a.m. and 8:30 a.m.	At the end of the second and third shift, System 1 copies the updated database to the multiported disk.

The next working day, these steps are repeated, with System 1 updating the on–line database and System 2 using a copy of the database for program development.

This arrangement has several advantages:

- Maximum up time. Most of the time, the backup database is not open. If System 1 fails while the backup database is closed, System 2 only needs to initialize the multiported disk and bring up its copy of the application. One or two data entry operators can walk over from System 1 to continue interactive operations, if needed, and database operations can continue. The database is current up to the last update — never more than one shift old.

  In the worst case, if System 1 fails while updating the database, the site can recover using the updates. Even so, System 2 can take over relatively quickly. The primary system doesn't need to be shut down for its database to be backed up.

- Database integrity. The primary database doesn't need to be used for program debugging. As new programs are written, programmers test them with the database on DB2. After testing, the programs are brought into the primary application, running on System 1.

- Current test base. Programmers have a nearly current copy of the database for developing and debugging their programs. Even if they destroy the database, they can restore it easily from either the backup tape or the multiported disk.

Using Mirrored LDUs

Another technique for maintaining high data availability is mirroring.

Mirroring causes the operating system to maintain two or three logically identical images (copies) of an LDU. With two images of an LDU, the operating system continues to run on the remaining image if one image is taken out of service (during backup, for example) or goes bad (perhaps suffers a hardware error).

If an LDU is mirrored, and one of the disk units that contains a mirrored image fails for any reason, the operating system will automatically transfer control to the good disk unit, providing continuous processing without interruption.

Under AOS/VS all mirrored images are said to be *hardware* mirrored. Hardware mirroring requires disk units that are the same model and are on the same disk controller.

AOS/VS II lets you use either *hardware* mirroring (where the images must be on identical model disk units on the same controller) or *software* mirroring (where the images can be on any model disk unit on any disk controller). You specify which disks units will be mirrored via the Disk Formatter (AOS/VS) or Disk Jockey (AOS/VS II) utility when you format the disks and create the LDUs. You tell the system to use mirroring by issuing the CLI INITIALIZE and MIRROR commands.

Mirroring, whether hardware or software, offers both advantages and disadvantages. When there has been a system failure and ESD did not or could not run, or if there has been a power failure, both hardware and software mirroring can take a long time to resynchronize mirrors. These situations aside, you may want to consider using LDU mirroring on your system.

The following section discusses hardware mirroring as it is implemented under AOS/VS and AOS/VS II.

Hardware Mirroring (AOS/VS and AOS/VS II)

Where hardware mirroring is possible — R.A.M.S. (rapid access mass storage) disks, 354–, 592, and 862–Mbyte disks, and any disk connected to an MRC I/O bus — the major advantage is synchronization speed. Hardware mirroring, particularly with R.A.M.S. disks, can provide faster synchronization than software mirroring (a few minutes versus several hours). With other disk types, the two forms of mirroring take about the same amount of time. The overhead needed to maintain synchronization (after the images are synchronized) is roughly similar for both hardware and software mirroring.

Another advantage of hardware mirroring is that the disk controller optimizes each read operation by reading from the disk where the read/write head is closer to the data.

The major disadvantage of hardware mirroring is that it depends on one disk controller. Hardware mirroring requires the disk units containing mirrored images to be accessed by the same disk controller. If this disk controller fails, all images will become inaccessible. The disk controller represents a single point of failure.

Another disadvantage is that hardware mirroring restricts you to using the same model disk for each disk unit containing a mirrored image.

In hardware mirroring, disk microcode writes the same data to two images of an LDU. Hardware mirroring creates a physically identical copy of the LDU. These images must be located on physical disks that are the same size, the same type, and on the same disk controller.

Figure 10–2 shows you an example of hardware mirroring with two images.

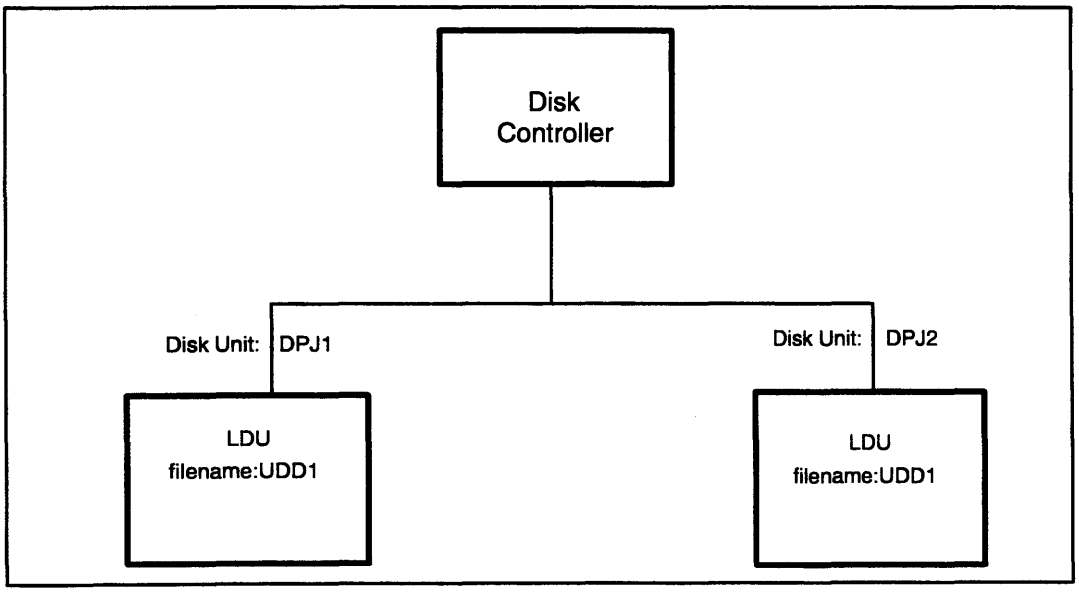


Figure 10–2 Example of Hardware Mirroring (with Two Images)

Disks pieces (portions of an LDU) using hardware mirroring must be the same size and on the same controller. You can mirror any LDU except an LDU named BOTH (used for SWAP and PAGE directories). When formatting a disk that will use hardware mirroring, make the LDU name and bad block table (BBT) entries of each image the same. (If you are running AOS/VS II, Disk Jockey will automatically take care of locating and sizing the BBT and remap areas.) For hardware mirroring (and software mirroring, if you are running AOS/VS II) the unique image ID must be different.

Figure 10–3 shows a typical case of hardware mirroring. This site has a model 6237 disk drive (three 354–Mbyte disks on one controller). DPJ0 is the system disk. DPJ1 and DPJ2 are mirrored; each holds an identical image of the LDU named UDD1.

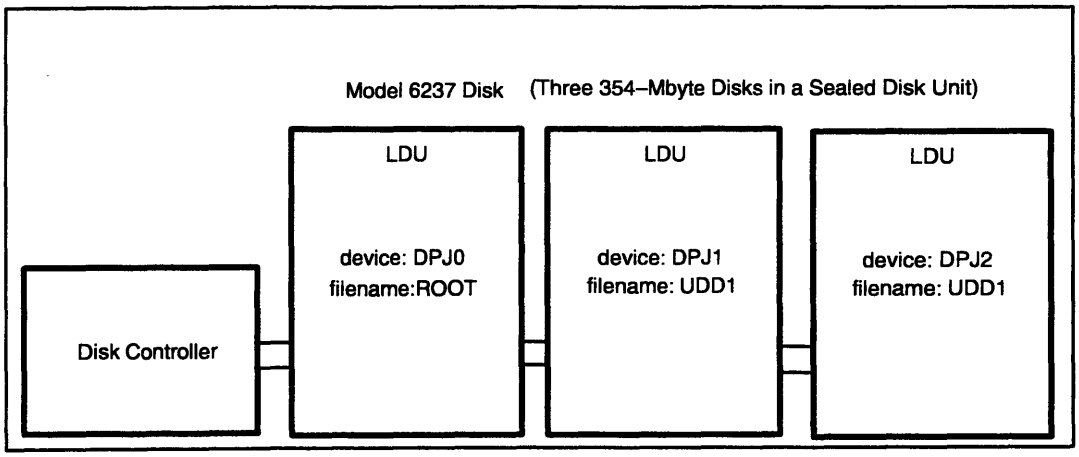


Figure 10–3 Hardware Mirroring with a Model 6237 Disk Unit

Hardware restrictions limit hardware mirroring to two images. If you choose to create three images of an LDU with hardware mirroring, the third image will be software mirrored.

Using Backup Disk Controllers and Mirroring (AOS/VS II Only)

A third technique for maintaining high availability is to use backup disk controllers along with disk mirroring. You can use a backup disk controller with each set of mirrored disks to achieve some measure of high availability. Using multiple disk controllers allows you to boot a different system in the event that a disk controller fails.

Figure 10–4 shows mirrored system and nonsystem LDUs attached to standard and backup controllers in a way that allows software or hardware mirroring. As with the previous figures, the mirrored LDUs are the root (:), UDD1, and CEO.

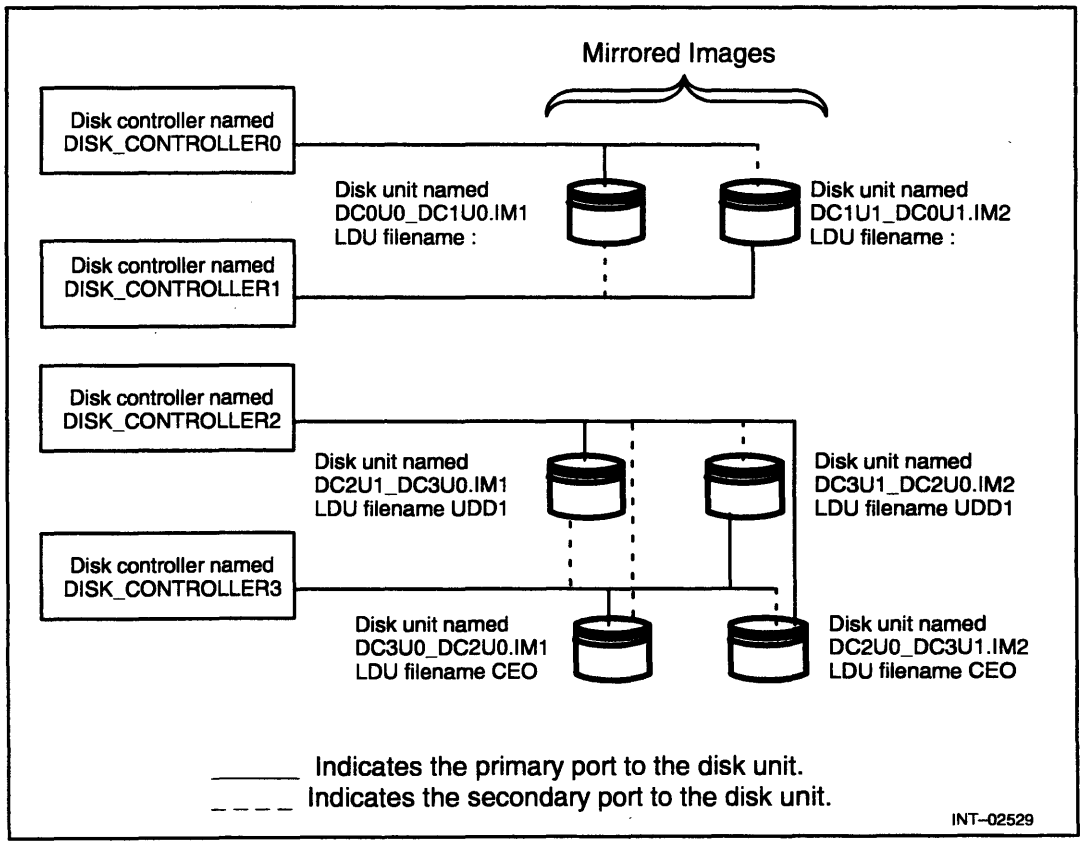


Figure 10–4 Software Mirroring Using Backup Disk Controllers

This arrangement provides the high availability of software mirroring and some measure of high availability even if a controller fails. If any disk unit or controller fails, the AOS/VS II system will break the software mirror and continue running on the other image. After a controller fails, the operator can (at a time convenient to applications) run a different AOS/VS II system, maintain LDU mirroring, and still access all LDU images. After the failing component is replaced, the operator can synchronize quickly using hardware mirroring (because the primary and secondary port to each unit allows hardware mirroring); then the operator can release the synchronized mirror and start the multiuser environment with software mirroring as usual.

The standard operating system runs with disk controller DC0 running the root LDU images; disk controllers DC1 and DC2 run the UDD1 and CEO images. There are two backup systems, stored with the standard system on the root LDU. One backup system, perhaps named IF_DC0_OR_DC2_FAILS.PR, uses disk controller DC1 for the root and UDD1 images and disk controller DC3 for the CEO images. The second backup system, which might be named IF_DC1_OR_DC3_FAILS.PR, uses disk controller DC0 for the root and UDD1 images and disk controller DC2 for the CEO images.

If a disk unit fails, the operating system will automatically stop mirroring with that unit and continue running on the other image only. If the disk named Disk_Unit_00 fails, the operating system will continue running on the other image, but it won't be bootable from that other image (Disk_Unit_00.MI) unless the disk controllers are in an MRC, which lets you boot from any disk unit, or the system operator manually changes the disk unit number from 1 to 0. In any case, a message on the system console will notify you so you can plan repair or replacement of the failed disk unit.

If a disk controller fails, the recovery action taken by the operating system will depend on which controller it is. If the system disk controller (DC0) fails, AOS/VS II will stop with a fatal error. AOS/VS II may or may not try to reboot, depending on the VSGEN setting selected for the automatic reboot parameter. In any case, the reboot will fail because the disk controller is defunct, but you can then start either of the two backup systems.

If a nonsystem disk controller (DC2 or DC3 in the previous example) fails, users with directories on the pertinent LDUs will lose access to the system. Hard error messages will be displayed on the system console. You can shut down the current system and then start the appropriate backup operating system.

Regardless of the failing controller, the UP.CLI macro (which initializes nonsystem LDUs and then starts mirroring) can be the same for the primary and the backup operating systems since the controller–to–unit route is part of the AOS/VS II system. The controller–to–unit route is not established with the INITIALIZE or MIRROR command. How to plan for and establish controller–to–unit routes is explained in the manual *Installing, Starting, and Stopping AOS/VS II*.

In addition to hardware mirroring, AOS/VS II offers software mirroring.

Software Mirroring (AOS/VS II Only)

The major advantage with software mirroring is higher availability. Since images can be on disk units managed by different disk controllers, the controller need not be a single point of failure. Software mirroring can provide continuous processing if either a disk or a disk controller fails.

Another advantage of software mirroring is flexibility; you do not have to use the same disk model for all disk units containing mirrored images.

Except when synchronizing R.A.M.S. disks (where hardware mirroring is significantly faster), synchronization time for hardware and software mirroring is roughly similar.

AOS/VS offers only hardware mirroring. AOS/VS II offers hardware mirroring as the default. AOS/VS II will use hardware mirroring on an LDU whenever possible.

If AOS/VS II cannot use hardware mirroring on an LDU for any reason, Disk Jockey will automatically use software mirroring.

Whether AOS/VS II does hardware or software mirroring is transparent to the user. You can override the hardware mirroring default and purposely create a software mirror by using the /NOHARDWARE switch on the MIRROR command, like this:

```
) MIRROR/NOHARDWARE/SYNC UDD1 @DPJ2 ↵
```

Unless you specifically override the default with this /NOHARDWARE switch, AOS/VS II will create a hardware mirror of the first two images. If you specify three images, the first two will use hardware mirroring, and the third image will use software mirroring.

With software mirroring, the operating system, rather than microcode, writes the data to duplicate images of an LDU. Software mirroring creates two (or three) logically, but not physically, identical images of an LDU. (For example, the bad block table (BBT) may or may not be in the exact same location on each image, and each image of the BBT may or may not contain the same entries.)

These images can be on physical disks of different types, and these disks can be on different controllers. Software mirrored LDUs, like hardware mirrored LDUs, do not have to be the same size as a physical disk. Mirrored images must be the same size, and you must make the LDU filename the same for both images. However, the unique image IDs must be different.

If your LDUs are not connected for hardware mirroring (the images you want to mirror are not on units connected to one controller), you must use software mirroring. Figure 10–5 shows software mirroring with three LDUs. The mirrored LDUs are the root (:), UDD1, and CEO.

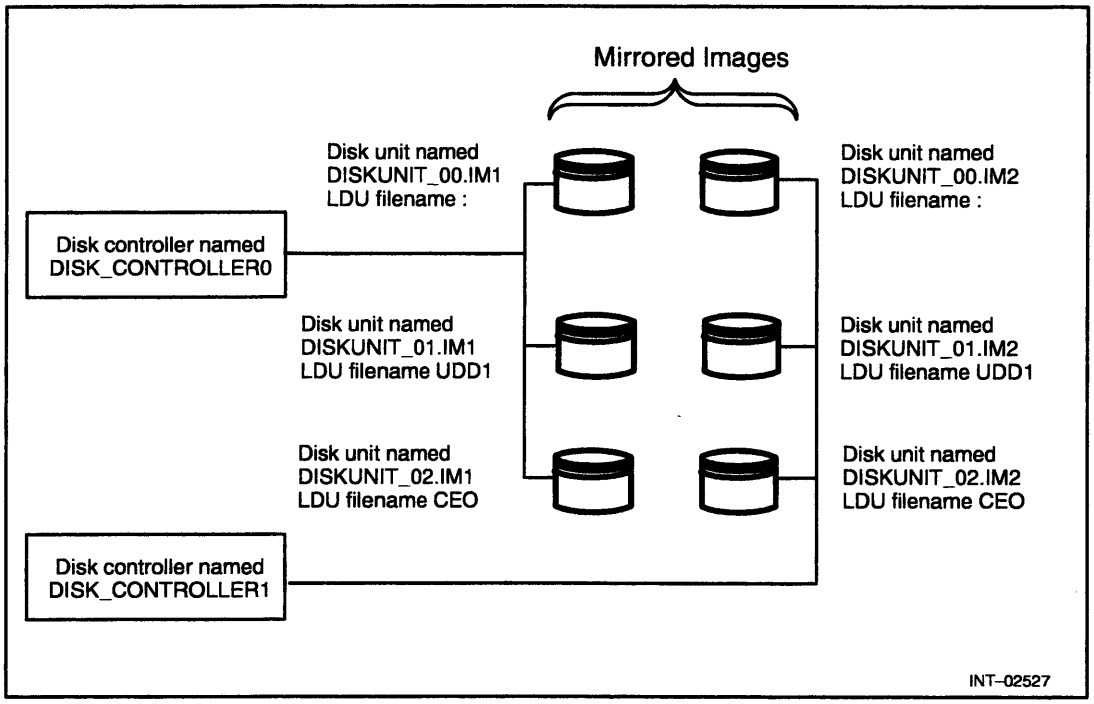


Figure 10–5 Software Mirroring Using Disks on Different Controllers

(The device names shown are hypothetical. Typical device names for MRC devices are MRCDISK_CONTROLLER_000E and MRCDISK000E00; typical ECLIPSE–bus device names are DPJ_CONTROLLER_0 and DPJ0.)

Software mirroring allows any pair of disks, regardless of class, to be mirrored. The only restrictions that apply to software mirroring are that the LDU filename must be the same for all images, the images must be the same size, and the unique image ID must be different for each image.

Using MRC Configurations (AOS/VS II Only)

If you have an MRC I/O system, it accepts backup disk controllers just as a non–MRC (ECLIPSE bus) system does.

For a system with an MRC subsystem and second MRC channel, software mirroring using separate controllers has the added advantage of high channel availability. If an MRC channel fails, AOS/VS II will continue using the good channel, its disk controller, and set of images.

Figure 10–6 shows a system with host computer, two MRC channels, MRC subsystem, disk controllers, and disk units. The controllers, disk units, and LDUs are the same as in previous figures.

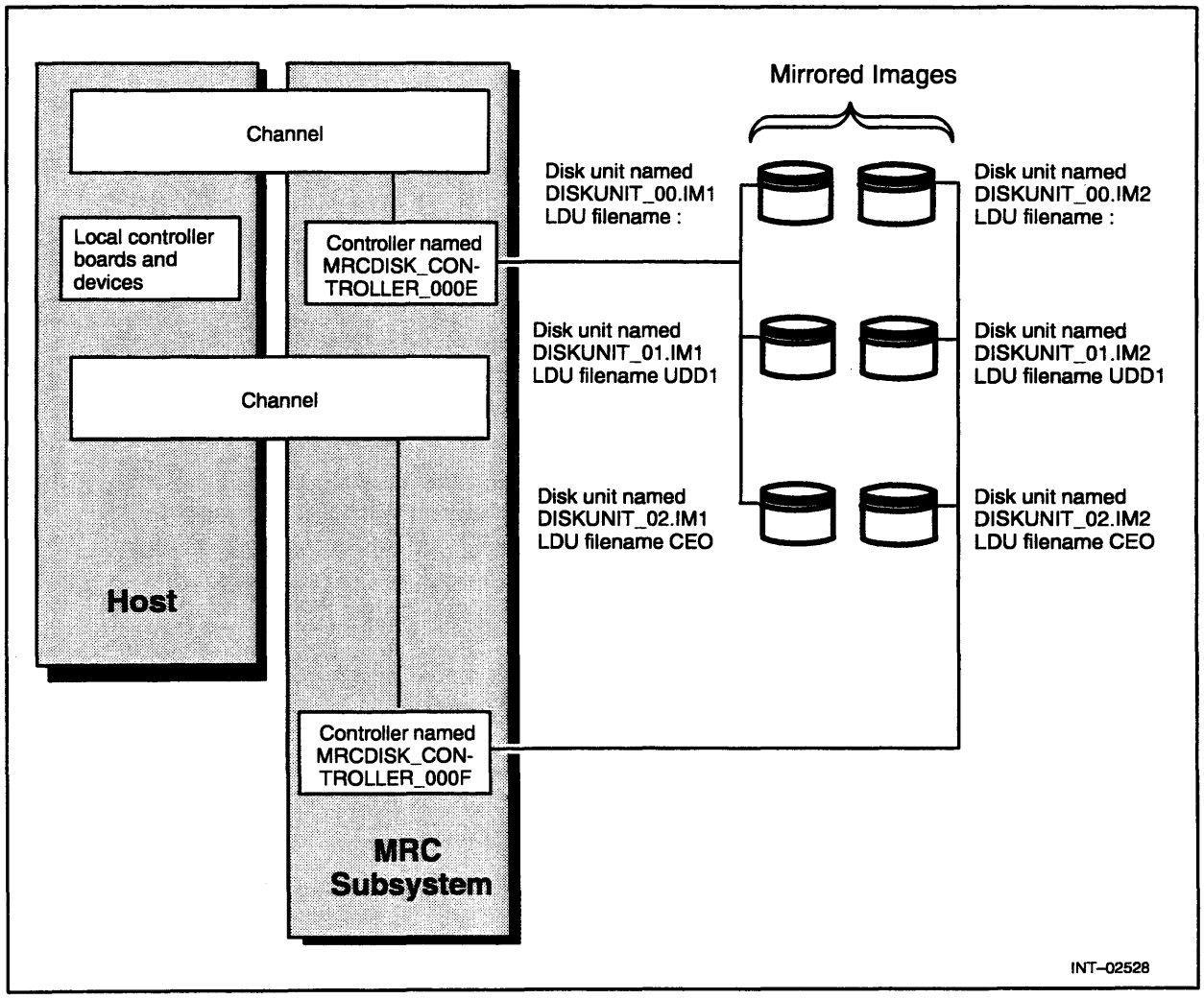


Figure 10–6 High–Availability System with MRC, Multiple Channels, Multiple Controllers, and Software Mirroring

In Figure 10–6 the controller named MRCDISK_CONTROLLER_000E handles disk units MRCDISK0000E00 (the root), MRCDISK0000E01 (UDD1), and MRCDISK0000E02 (CEO). MRCDISK_CONTROLLER_000F the second disk controller, which comes off a separate MRC channel, controls the second AOS/VS II system disk units MRCDISK0000F00 (the root), MRCDISK0000F01 (UDD1), and MRCDISK0000F02 (CEO), which are mirror images of the disk units on the first disk controller.

In summary, hardware and software mirroring, with or without multiple disk controllers and multiported disk units, can increase system availability by making backup units with current information on them available in the event any disk unit fails. Configuring these systems is explained in the manual *Installing, Starting, and Stopping AOS/VS II*.

For information on formatting mirrored LDUs under AOS/VS, see the chapter on the Disk Formatter in the manual *Installing, Starting, and Stopping AOS/VS*.

Using CLARiiON or H.A.D.A./MV Storage Systems

The CLARiiON disk–array storage system, available with AOS/VS or AOS/VS II, and the H.A.D.A./MV (High–Availability Disk–Array) storage system, available only with AOS/VS II, provide a compact, high-capacity, high-availability bank of disk storage for one or two computer systems. The CLARiiON disk array offers up to 40 Gigabytes (Gbytes) of disk storage or 32 Gbytes of high–availability disk storage. H.A.D.A./MV offers up to 30 Gbytes of disk storage or 24 Gbytes of high–availability disk storage. Each uses up to 30 disk modules that you can replace under power. H.A.D.A./MVs can also support two cartridge tape drives that you can replace under power. These storage systems provide the high data availability of mirroring at lower cost, and without the cost of duplicating 100% of the data.

The CLARiiON tape–array storage system, available only with AOS/VS II, can provide high data availability during backups and restorations.

The H.A.D.A./MV connects to the computer via one or two SCSI–2 (small computer system interface) buses; the computer runs disks in the subsystem through one or two DPJ–type disk controllers as if the disks were standard disk units.

The CLARiiON disk–array system connects to an AOS/VS system via an ECLIPSE bus or to an AOS/VS II system via an MRC SCSI–2 adapter or via one or two SCSI–2 (small computer system interface) buses. The computer runs disks in the subsystem through one or two DPJ–type disk controllers as if the disks were standard disk units.

For more information about configuring and operating these systems, see *Configuring and Managing the High–Availability Disk–Array (H.A.D.A./MV) Subsystem*, *The CLARiiON™ Series 2000 Disk–Array Storage System with AOS/VS*, *The CLARiiON™ Series 2000 Disk–Array Storage System with AOS/VS II*, and *The CLARiiON™ Tape–Array Storage System with the DG/UX or AOS/VS II Operating System*.

Forcing the Release of an LDU (AOS/VS II Only)

On occasion, you may be unable to release an LDU (as at shutdown or after a RELEASE command). The system will report

LDU in use, cannot release

The problem may involve a hard error that prevents normal release; if so, you will also see a *HARD error* message. Or, more likely, it may involve a logged–on user whose working directory or search list includes the LDU. Whatever the problem, you can force the release of such an LDU by using the RELEASE command with the /FORCE switch. RELEASE/FORCE releases an LDU and any subordinate LDUs, without regard for I/O errors or users who are using the LDU. Privilege requirements are the same as for a normal release: Write and Execute access to the directory in which the LDU was initialized and Owner and Execute access to the LDU — or Superuser on. The /FORCE switch is available in CLI32 only.

CAUTION: Releasing an LDU closes all files on the LDU and all subdirectories, including LDUs if any were initialized into the LDU. A user who has files open on any force–released LDU will lose all changes — including any modified shared pages — that have not yet been written to disk. For example, a user editing a text file with the SED editor will lose all changes he/she made since the last BYE or SAVE command. Therefore, be sure to warn users if possible before forcing the release an LDU. (This caution is irrelevant if a hardware error prevents access to the LDU, since users will lose their changes no matter what you do.)

If mirror synchronization is in progress, the synchronization procedure will be aborted. (But if an LDU is mirrored and the images are synchronized, forcing release will not break the mirror.) You cannot release the root directory (:) or an LDU containing the SWAP or PAGE directory.

After you force the release an LDU, you can proceed with the operation you had planned, for example repair of a faulty disk unit.

After you force a release, any process that tries to access a file on the released LDU will receive the error message *Directory not available because the LDU was force released.* A process whose search list includes a directory in the released LDU will receive the same error message when it tries to access files. If a process receives this error, the process must manually respecify its search list to omit all directories on the released LDU. If the process' initial working directory was on the LDU, the process should change to the root directory and then to another valid directory before trying to continue processing. When that LDU is next initialized, the user process must issue a new SEARCHLIST command to restore the LDU to his/her search list.

Forcing release does not close channels to files open on the released LDU. Thus, to resume normal processing with a file whose parent LDU has been released after the LDU is next initialized, the user process must close and then reopen the file. An easy way to do this is to exit from and then re–enter the application that was running when the release occurred.

Using the Runtime Configuration Manager (AOS/VS II and MRC Only)

If a disk or tape controller on a MRC subsystem fails, AOS/VS II will try to switch to an alternate route to the unit(s) on that controller. When it is successful, AOS/VS II sends a message to the system console and to the system error log file and will not switch back to the primary route until the controller has been fixed and you restore the primary route. You can restore the primary route by shutting down the system. As part of the high–availability implementation, however, you can use the Runtime Configuration Manager (RCM) to restore the primary route without shutting down your system.

RCM is a menu–driven utility with keywords and on–line Help messages. You can use RCM to display the current MRC routes or reset MRC routes to their original paths. For more information, see *Installing, Starting, and Stopping AOS/VS II*.

Using Automatic Dump and Automatic Reboot (AOS/VS II Only)

With AOS/VS II only, you can combine the ability to dump the system's memory to a user–defined system area on a DPJ or MRC type disk, the related ability to dump the system's memory automatically, and the ability to reboot the system automatically. Taken together, these features provide high availability after a panic occurs.

Dumping to a system area is much faster than dumping to tape. For example, dumping 256 megabytes of memory from an ECLIPSE MV/30000 computer to a 2–gigabyte cartridge tape takes approximately 18 minutes. Dumping the same data to a system area on a Model 6621 disk can take less than four minutes. After the system reboots itself, you can dump the memory dump in the system area to tape (using DUMP_II to dump the contents of system areas), or a Data General representative can analyze the system dump while it is still in the system area.

You can enable automatic dumping of a system's memory, choose a default dump device, and enable automatic reboot through VSGEN host parameters. With automatic dumping, all dump questions time out and the dump proceeds automatically to the default dump device; an operator does not have to be present. You can choose either a tape unit or a disk unit's system area as the default dump device. If you choose to make a tape unit the default dump device, you should choose a tape unit that can hold the entire dump. Otherwise, if the dump requires multiple volumes, it will stop and wait for an operator to change the tape. With automatic reboot, the system automatically reboots after the dump completes.

For more information, see the manual *Installing, Starting, and Stopping AOS/VS II*.

Automatic IAC Reboot

Beginning with AOS/VS II Revision 2.10, the operating system can automatically reboot a failed IAC, which helps improve system availability. The following information is repeated from *Installing, Starting, and Stopping AOS/VS II*.

If a fatal error occurs in a terminal controller, AOS/VS II will display on the system console a message of the form:

From system at time on date:
Terminal controller panic,
Device code: dcode Engine: n (*dcode* is the device code. *Engine: n*
Error Code: e identifies the processor; it appears only if the controller has multiple processors.)

Users on terminals connected to the failed controller will experience no response. AOS/VS II will terminate their inactive processes.

Then AOS/VS II will try to copy the failed controller's memory to a disk file in system directory :TSDUMPS. On success, it will display

From system at time on date:
Terminal controller dump file created.
File :TSDUMPS:TS.dcode.n.dd_mon_yy.hh_mm_ss

where

dcode is the device code;
n is the processor number on multiple–processor controllers;
dd_mon_yy.hh_mm_ss represents date and time.

See the section "If the Problem is With a Terminal Controller" in Chapter 10 for information about submitting an STR with this dump file.

Then AOS/VS II will try to reboot the controller. The number of times it will try was specified at VSGEN; the default number given by VSGEN is 3. If the reboot operation succeeds, AOS/VS II will display a message of the form

From system at time on date:
Terminal controller rebooted
Device code: dcode Engine: n (Again, *Engine: n* appears only if the controller has multiple processors.)

Lines on the rebooted controller will then be usable. Directly connected lines that were enabled by EXEC at the time of the failure will remain enabled after the reboot. On directly connected lines EXEC will display the logon banner, and users can log on.

If AOS/VS II cannot reboot the controller, it will display a message of the following form and then continue without the controller. Later, at shutdown, the system will act on this error condition by displaying a panic (*fatal error*) message.

From system at time on date:

Terminal controller { *has exceeded reboot limit* / *failed reboot* }

Terminal controller has been removed from service.

Device code: dcode *Engine: n* (Again, *Engine: n* appears only if the controller has multiple processors.)

End of Chapter

Chapter 11
Using Other Runtime Tools

Read this chapter when you want to use one or more runtime tools to help operate the system. This chapter describes them in the following major sections:

- Using the OP or Master CLI
- Locking the OP or Master CLI
- Viewing and Comparing Files (BROWSE, DISPLAY, FILCOM, and SCOM)
- Displaying the Process Environment (PED)
- Logging Operating System Events (ERROR_LOG, SYSLOG, Superuser logging, and CON0_LOG)
- Reporting Operating System Events (REPORT)
- Testing System Confidence (CONTEST)
- Monitoring ECLIPSE–bus Disk I/O (DISCO)
- Displaying AOS/VS II LDU and Disk Information (LDUINFO)
- Changing Program Preamble Parameters (SPRED)

Using the OP or Master CLI

The CLI — the operator interface to the operating system, is a very powerful tool, with over 100 commands and a flexible macro facility that allows you to create your own commands.

By default, when the operating system comes up, it runs a CLI process on the system console. This CLI has the process ID (PID) 2, has the username OP (operator), and it is the father of all lower processes. It has all privileges, including System Manager, Superprocess, and Superuser. This CLI is called the master CLI.

The master CLI can set the system date and time; it can start and stop system logging; it can run the UP.CLI and DOWN.CLI macros; and it can terminate any and all processes without turning Superprocess on.

The UP.CLI macro issues a number of commands to the master CLI, including an ACL command that gives users access to tape units, a PROCESS command that starts EXEC, and finally an EXECUTE command that starts a son CLI under the master process. This son CLI is usually the one that runs on the system console.

The son CLI can be a standard CLI or a locked CLI (to safeguard the system console). You choose the type of CLI you want in the UP.CLI macro.

A standard son CLI running under the master has nearly all the privileges of its father. It has the username OP, which allows it to issue commands to EXEC. It also has the privileges that the system operator needs to run the system.

A locked CLI (either a 32–bit CLI that you've *locked* with a password or the 16–bit program LOCK_CLI), though, has no privileges and practically no power. It can issue commands to EXEC, but otherwise, it can do almost nothing, which makes it an ideal guardian of the system console. (When unlocked, any locked CLI has the same powers as a standard CLI.) See the next section, "Locking the OP or Master CLI," for more information.

To get back to the master CLI, a person needs to terminate the son CLI on the system console. Terminating a standard CLI is easy (type BYE and press NEW LINE). Terminating a locked CLI — whether you're running the 16–bit or 32–bit CLI — requires that you type a pre–assigned password.

Table 11–1 summarizes some process–oriented CLI commands and macros that can be issued by the master CLI, a standard son, or any user CLI that has appropriate privileges. It describes the commands briefly, in the context of system operation. For full syntax and all switches, of any CLI command, type HELP/V followed by the name of the CLI command, or see the CLI manual.

For a session with the CLI, see the manual *Learning to Use Your AOS/VS System*.

You can abbreviate any CLI command to the shortest string of characters that uniquely identifies the command.

Table 11–1 Operator–Oriented CLI Commands and Macros

Command or Macro	Description	Example
?.CLI	A macro that describes all processes on the system.	) ? ↵
ACL	Displays or sets a file access control list. ACL is the primary access control tool, further described in Chapter 14, "Maintaining System Security."	) ACL/V :FORMS ↵) ACL OP,OWARE ↵
BROADCAST.CLI	A macro that sends a message to all user terminals.	) BROADCAST Log off! ↵
BYE	Terminates the current CLI. The process' father (the process from which it was executed) gets control. If there is no father, as with the master CLI, BYE starts a system shutdown. Typed to a user CLI process, BYE logs the user off.	) BYE ↵
CHARACTERISTICS	Displays, sets, or resets the hardware characteristics of a terminal, overriding the VSGEN setting if desired. Use it when you want to change behavior of the system console or a user terminal.	) CHAR/PM ↵ ...) CHAR/RESET ↵) CHAR/DEF/8BT ↵
CONTROL or CX	Sends an IPC message of commands to process in :PER. Use it to control EXEC and other Data General products. The CX macro expands to CONTROL @EXEC.	) CX ALIGN ↵
COPY	Copies a source file to a destination file. COPY is useful when you want to copy a file literally, without the header information that DUMP or DUMP_II writes.	) COPY @MTB0 TBOOT ↵) COPY @NULL @MTB0 ↵

(continued)

Table 11–1 Operator–Oriented CLI Commands and Macros

Command or Macro	Description	Example
CREATE	Creates the file or pathname you specify. Use /MAX=n to create a control point directory of n blocks, /DIR to create a standard directory, and /LINK to create a link entry to a destination file. To insert text in the new file, use CREATE/I.	) CREATE/MAX=80 FOO ↵) CRE/I FOO ↵
DELETE	Deletes the file(s) given in template. You will often delete files in the course of your system housekeeping.	) DELETE/V +.BRK ↵) DEL/V DIR:XX+ ↵
DIRECTORY	Displays the working directory name or sets the directory you specify as the working directory.	) DIR ↵) DIR :UTIL ↵
DOWN.CLI	A DG–supplied macro that brings down EXEC.	) DOWN ↵
DUMP file	Copies files to tape or disk, most often for backup. In CLI32, DUMP is a macro that executes DUMP_II.	) DUMP/V TAPE:FOO ↵
FILESTATUS	Describes files in any directory. It's one of the most useful commands.	) FILES/AS ↵) F/AS/S :+ ↵
HELP HELPB	Display the CLI topics for which on–line help is available. With a CLI command as an argument, gives help on that topic. HELPB uses the BROWSE utility to display the Help file.	) HELP ↵) HELP/V ACL ↵) HELPB FILESTATUS ↵
INITIALIZE	Grafts a logical disk made up of one or more physical disks (AOS/VS), or part of one or more physical disks (AOS/VS II), into the working directory. For a mirrored LDU, use an exclamation point (!) to separate the LDU images. Use /NOMIRROR to initialize one image of a mirrored LDU.	*Su*) INIT @DPJ0 ↵ *Su*) INIT @DPJ1!@DPJ2 ↵ *Su*) INIT/NOMIR @DPJ2 ↵

(continued)

Table 11–1 Operator–Oriented CLI Commands and Macros

Command or Macro	Description	Example
JPINITIALIZE	Initializes a job processor, n, other than the default job processor (the default processor is usually 0). This command gives the operating system control of the new job processor. You need the System Manager privilege to use this command, which only works on systems that have more than one job processor. If you omit switches, the system loads microcode from the default microcode file into the job processor, whether or not microcode is already loaded. If microcode has already been loaded into the job processor, use the switch /EXISTING, which tells the operating system to use existing microcode. This saves time on warm starts. The operating system will report an error if microcode hasn't been loaded. To load a nondefault microcode file (instead of loading the default or using the existing microcode), use the switch /MCODEFILE=pathname. Normally, you will want to insert the JPINITIALIZE command in your UP.CLI macro, before any PROCESS or XEQ commands.	) JPINITIALIZE 1 ↵
JPRELEASE	Releases an initialized job processor other than the default processor on a system with more than one job processor. You can use this command if you wanted to remove a job processor from the system without shutting down. If you've defined a logical processor via the optional CLASP program or other user program), and the job processor you want to release is the last one connected to your logical processor, include the /LAST switch. (You will get an error if you don't include it.)	) JPRELEASE 1 ↵

(continued)

Table 11–1 Operator–Oriented CLI Commands and Macros

Command or Macro	Description	Example
LOAD	Loads files backed up with DUMP or DUMP_II. In CLI32, LOAD is a macro that executes LOAD_II.	) LOAD/V TAPEFILE ↵) LOAD/V @MTB0 ↵
MIRROR	Initializes a second image of a mirrored disk and begins synchronization. Use /SYNC when the image that is initialized (via /INITIALIZE/NOMIRROR) is the preferred image. Use /FORCESYNC when the image you are bringing in is more recent than the initialized image. Using /WAIT with either /SYNC or /WAIT suspends the CLI until the synchronization has finished, which can take several hours. Use /WAIT only if you need to ensure that the mirror is synchronized before the system comes up.	*Su*) MIRROR/SYNC & ↵ *Su&*)UDD1 @DPJ1 ↵
MOVE	Copies file(s) named in template to another directory. You'll use MOVE often for housekeeping. See also RENAME.	) MOVE/V :ERMES ↵
OFF.CLI ON.CLI	Macros that turn Superuser and Superprocess on and off. Typing ON or OFF is easier than typing Superuser ON or Superuser OFF.	) ON ↵ *Su*) OFF ↵) ON/P + ↵
POP	Restores the previous CLI environment. To descend to the next CLI environment, use PUSH.	) PUSH ↵) CHAR/PM ↵) DIR XX:YY ↵ ...) POP ↵
PRIVILEGE	Lets you set or display the current privileges. The valid arguments are Superprocess, Superuser, and System Manager. If no argument is supplied, the CLI displays the current privileges (if any).	) PRIVILEGE ↵) PRIV SUPERUSER ↵

(continued)

Table 11–1 Operator–Oriented CLI Commands and Macros

Command or Macro	Description	Example
PROCESS	Creates a new process, as XEQ does, but is more versatile. You can write PROCESS commands for your applications into the UP.CLI macro.	) PROC/IOC=@CON2& ↵ &)/DEF/BLOCK :CLI ↵
PUSH	Descends to a new CLI environment. To ascend, use POP.	See the POP example.
QBATCH	Posts a batch job; you may use batch for prolonged operations that can run by themselves.	) QBATCH BACKUP ↵
QDISPLAY	Describes all queues and their status; this command is handy in a multiuser environment.	) QDISPLAY ↵) QD ↵
QPRINT	Posts a printing job; you use this when you want hardcopy.	) QPRINT MYFILE ↵
RENAME	Renames a file or (CLI32 and AOS/VS II only) transfers the file to a new directory. Use to preserve a backup file when you know that the program you're about to run will delete the original.	*Su*) REN ERMES & ↵ *Su&*) ERMES.OLD ↵ *Su*) REN ERMES & ↵ *Su&*):OP:ERMES ↵
RUNTIME	Describes process statistics; use it when you think a process may be malfunctioning. If the process is using all the CPU time, you know it's malfunctioning.	) RUN ↵) RUN 1 ↵
SEARCHLIST	Displays or sets your search list, a list of directories the CLI scans when it can't find a file in the the working directory. The UP.CLI macro sets your search list, but you may want to change it for some system operations.	) SEARCH ↵) SEA [!SEA] :NEW_DIR ↵
SED.CLI	A macro that executes the SED text editor.	) SED AFILE ↵

(continued)

Table 11–1 Operator–Oriented CLI Commands and Macros

Command or Macro	Description	Example
SEND	Sends a message to a process ID (PID). You'll use this often to communicate with users.	) SEND 20 LUNCH? ↵
SPACE	Tells you how much disk space is used, and how much remains free, in a control point directory (CPD) (or any directory with CLI32 and AOS/VS II). Use with CPD to set a different maximum size of n blocks.	) SPACE : ↵) SPA :UDD ↵) SPA :UDD:LOU 5000 ↵
SUPERPROCESS	Turns Superprocess, the power to control any process, on or off. You need this privilege to terminate a brother or father process.	) SUPERPROCESS ON ↵ *Sp*) SUPERP OFF ↵)
SUPERUSER	Turns Superuser, the power to access any file on or off. You may need this privilege to create, edit, or delete files outside your directory.	) SUPERU ON ↵ *Su*) SUPERU OFF ↵)
SYSTAPE.CLI	DG–supplied macro that creates a tailored system tape.	*Su*) SYSTAPE @MTB0 ↵
TERMINATE	Terminates a subordinate process. This is used in the DOWN macro; you will use it yourself when you need to terminate a process. Often used with SUPERPROCESS ON.	) TERM 34 ↵
TYPE	Types an ASCII file on the terminal. Use it to read a file or tape label.	) TYPE FOO ↵) TY @MTB0 ↵
UP.CLI	DG–supplied macro that brings up EXEC and the multiuser environment.	) UP ↵
WHO	Displays the username associated with a process (PID n). Use it to find out who is running a process.	) WHO ↵) WHO 30 ↵
WRITE	Displays arguments on the terminal or writes them to a file.	) WRITE [!DATE] ↵
XEQ	Executes a program.	) XEQ VSGEN ↵

(concluded)

Filename Templates

A filename template includes a special character that specifies a set of filenames. The most common template characters are as follows:

Character	What It Means
*	Match any single character except a period.
–	Match any series of characters not containing a period.
+	Match any series of characters, including periods.
\	Omit a series of characters.
#	Search the specified directory and all subordinate directories. Without this template, the search is restricted to the working (or specified) directory.

For example,

This command	Searches for filenames of
) FILES/AS ****** ↵	six characters without a period.
) FILES/AS **.CLI ↵	two characters without a period, ending in .CLI.
) FILES/AS –↵	any characters without a period.
) FILES/AS –.CLI ↵	any characters without a period, ending in .CLI.
) FILES/AS + ↵	any characters.
) FILES/AS +.CLI ↵	any characters ending in .CLI.
) FILES/AS :#:+ ↵	any characters in all directories in the system, starting at the root.
) FILES/AS :#:+.SR ↵	any characters ending in .SR in all directories, starting at the root.

Templates work with most CLI commands and are extremely useful. For file searches, especially prolonged ones with #, you can apply the /L=@LPT switch, so the listing will show you the pathnames of the files found.

Filename Suffixes and Their Meanings

Certain filename suffixes indicate the kind of material in the file. These suffixes are useful because they tell you at a glance what's in a file. You may want to delete files with certain suffixes periodically to conserve disk space. The most common filename suffixes are listed in Table 11–2.

Table 11–2 Common Filename Suffixes

Suffix	Meaning
.BRK	This is a break file created by a program on abnormal termination. These are occasionally useful in problem diagnosis, but generally you can delete them.
.CFG	AOS/VS II only. This is a sizer file created by Disk Jockey. (Read the manual *Installing, Starting, and Stopping AOS/VS II* for more about .CFG files.)
.CLI	This is a CLI macro. It's in ASCII, and you can type it.
.CONFIG	AOS/VS II only. This is a VSGEN configuration file, used by VSGEN to generate a tailored system.
.CSF	AOS/VS only. This is a VSGEN customer specification file, which contains the specs for an operating system. You can type such files (usually in :SYSGEN) to see what hardware and software a system supports. A companion file, which VSGEN reads, has the .SSF suffix.
.DL .DS	These are dialog files created by the LINK program for Data General's SWAT debugger. The files are needed by SWAT; but after a program has been debugged, the .DL and .DS files should be deleted.
.ED	This is a SED edit file, created by SED for its own use during an edit session. Such files can be deleted if you need disk space.
.JOB	This is a batch input file, which the ?00028.CLI.001.JOB system deletes after the job finishes. Such a file remains in a user's working directory only if the job didn't finish normally. You can type these files.
.LB	This is a library file: a group of .OB files from which Link takes material it needs to build .PR files.
.LPT	This is a temporary print queue file, which the system deletes if the job finishes normally. These files are created in directory :QUEUE.
.MCF	This is a microcode file, which the system bootstrap program can load into the main CPU if microcode is not already loaded.
.MDM	This is a memory dump file, created by a ?MDUMP system call in a program. Such a file can help Data General solve problems; you should include a memory dump file if possible when you submit a Software Trouble Report. Memory dump files are rare.

(continued)

Table 11–2 Common Filename Suffixes

Suffix	Meaning
.NAMES	AOS/VS II only. This is a device names file, created by VSGEN, and used by Disk Jockey to obtain device names for your tailored system.
.OB	This is an object file, produced by compilation or assembly of a source file. LINK uses .OB files to build .PR files.
.OL	This is an overlay file, used by the .PR file of some utility programs.
.PR	This is an executable program file.
.SR	This is a source file, in ASCII. It can be a basis for an executable program, or it can assign parameters. You can type such files.
.SSF	AOS/VS only. This is a VSGEN system specification file, used by VSGEN to generate a tailored system. Delete the file only if the AOS/VS system based on it is obsolete.
.ST	This is a symbol table file, which gives symbols numeric addresses. LINK creates it automatically. The patch program and autopatch macros need these files. For a user program, unless the program may need debugging or patching, you can delete the .ST file.
.TM .TMP	This is a temporary file, created by a utility program for its own use. The utility that creates .TMP files usually deletes such files after normal termination. Aside from the VSGEN .TMP files, you should delete these.

(concluded)

Other suffixes are used for other Data General products. For example, .C denotes a C source file, .H a C include file, .F77 a FORTRAN 77 source file, .PL1 a PL/I source file, and so on.

Pushing and Popping

There are many levels of CLI available, each with its own environment. Normally, the CLI runs on level 0, the highest level.

You can use the PUSH command to move down a level and set up a new CLI environment with new characteristics, directory, search list, and others (described in the manual *Using the CLI (AOS/VS and AOS/VS II)*). When you're done with the new environment, use the POP command to return to the higher level. The LEVEL command tells you the current level. PUSH and POP can help eliminate confusion and save time. For example,

```
) DIR ↵
:UTIL                                  The working directory is :UTIL.
) PUSH ↵                               Push a level.
) DIR :OBS:ZZZ ↵                       Change the directory and other values.
) SUPERU ON; SEARCH ZZ:XX ↵            The semicolon allows you to stack
                                       CLI commands.

Su) MOV/V :UDD:OP FILEX ↵
FILEX
Su) POP ↵                              Pop to restore everything.
) DIR ↵
:UTIL
```

Superuser, Superprocess, and System Manager Privileges

The Superuser privilege, if on, allows a user to read, write, modify, delete, or change the ACL of any file on the system. Superusers can bypass all file access controls at will. This privilege can imperil all files. The CLI prompt is *Su)* when Superuser is on.

Superprocess, if on, allows a process to block any other process, become resident, or terminate any other process, including the master CLI, which would shut down the operating system. The CLI prompt is *Sp)* when Superprocess is on.

System Manager privilege, if on (CLI command PRIVILEGE), allows a process to set or change the system time, initialize or release job processors, start or stop system logging (although the 32–bit CLI is required for some functions), and otherwise control many system management functions. The CLI prompt indicating that the System Manager privilege is on is *Sm)*.

The CLI prompt is *SpSu)* when both Superuser and Superprocess are turned on. The CLI prompt when Superprocess, Superuser, and System Manager privileges are all turned on is *SpSuSm)*.

The Superuser, Superprocess, and System Manager privileges are options for each user profile. If you want a secure system, users should not have these privileges, nor should they have *Change username* privilege, which gives easy access to privileged usernames. In some systems, even the operator doesn't have these privileges — described in the next section.

Locking the OP or Master CLI

Because the OP or Master CLI has such power, it is a good idea to protect the system console, either by making sure it is locked in a computer room inaccessible to anyone but a trusted operator or by "locking" the CLI in such a ways that it will not obey commands that might compromise the system. Depending on whether you are running the 16–bit CLI or the 32–bit CLI, read the following sections that describe how to lock the CLI.

Locking the 32–bit OP or Master CLI

The locking technique for the 32–bit CLI is easy to use and very flexible. To enable locking, you first use the PASSWORD command to specify a password.

To specify a password, enter the 32–bit CLI, and type PASSWORD and then press NEW LINE. You will be prompted for a password of 1 – 32 alphanumeric characters. The 32–bit CLI will ask you to re–enter the password as a means of confirming the password. Now the 32–bit CLI has the password internally.

Now type PASSWORD/WRITE= and a filename. The 32–bit CLI writes out the (encrypted) password to the specified file. You should change the access on the file containing the password, as well as the directory containing the file, so that only the system manager and system operator(s) can read the file and learn the password.

Now edit your system's UP.CLI macro to include

PASSWORD/READ=*filename*
LOCK/FILE=*filename*

and insert that command immediately following these lines that says EXECUTE CLI. Then insert SUPERUSER ON before the EXECUTE line and insert the line SUPERUSER OFF after.

The 32–bit CLI will read the encrypted password every time you execute the UP.CLI macro.

By default, just like LOCK_CLI, while locked, a locked 32–bit CLI executes all but the following CLI commands:

BLOCK	ENQUEUE	MOVE	RELEASE
BYE	EXECUTE	PRIVILEGE	RENAME
CHAIN	INITIALIZE	PROCESS	SUPERPROCESS
CONNECT	JPINITIALIZE	PROMPT	SUPERUSER
COPY	JPRELEASE	QBATCH	TERMINATE
DEBUG	LOAD	QFTA	XEQ
DELETE	LOCALITY	QPLOT	
DUMP	MIRROR	QSUBMIT	

Additionally, the 32–bit CLI lets you lock individual CLI and EXEC commands. For more information, type HELP/V LOCK and press the NEW LINE key.

You cannot execute programs (including the LABEL program, which labels tapes) from a locked LOCK_CLI. LOCK_CLI also ignores the process termination sequences CTRL–C CTRL–B, CTRL–C CTRL–E, and CTRL–D CTRL–D.

While locked, LOCK_CLI allows printing (via QPRINT) of files in the root directory only, and it ignores the /L= switch.

Locking the 16–bit OP or Master CLI (LOCK_CLI)

LOCK_CLI is a lockable CLI that runs only under the 16–bit CLI, and requires a password to unlock or terminate. LOCK_CLI is designed to safeguard the system console from unauthorized people. It has two special commands, LOCK and UNLOCK. LOCK_CLI comes up in the locked state. While locked, it executes all but the following CLI commands:

BLOCK	ENQUEUE	MOVE	RELEASE
BYE	EXECUTE	PRIVILEGE	RENAME
CHAIN	INITIALIZE	PROCESS	SUPERPROCESS
CONNECT	JPINITIALIZE	PROMPT	SUPERUSER
COPY	JPRELEASE	QBATCH	TERMINATE
DEBUG	LOAD	QFTA	XEQ
DELETE	LOCALITY	QPLOT	
DUMP	MIRROR	QSUBMIT	

You cannot execute programs (including the LABEL program, which labels tapes) from a locked LOCK_CLI. LOCK_CLI also ignores the process termination sequences CTRL–C CTRL–B, CTRL–C CTRL–E, and CTRL–D CTRL–D.

While locked, LOCK_CLI allows printing (via QPRINT) of files in the root directory only, and it ignores the /L= switch.

You can unlock LOCK_CLI by typing UNLOCK and pressing NEW LINE and then typing the password and pressing NEW LINE; you can lock it again by typing LOCK and then pressing NEW LINE.

For example, assume the password is XYZZY:

) WHO ↵ *PID 20 : OP OP :LOCK_CLI.PR*	Check the process. It's LOCK_CLI.
) SUPERUSER ON ↵)	Try to turn on Superuser. No error, but no Superuser prompt.
) F/AS :UPD:+ ↵ *Warning: Read or write access is required, File :*	Try to check files in :UPD. Not allowed.
) BYE↵) WHO ↵ *PID: 20 OP OP :LOCK_CLI.PR*	Try to terminate LOCK_CLI. Check again. Still LOCK_CLI.
) UNLOCK ↵ *XYZZY* ↵	Start to unlock. Type password, which doesn't echo.
) SUPERUSER ON ↵ *Su)*	See if it's unlocked. Yes — it now has full CLI powers.
Su) LOCK ↵)	Lock it again. Typing LOCK also turns off Superuser and Superprocess.

You can type the password in either upper– or lowercase. LOCK_CLI is shipped with a password, but this password should be changed to protect system security. The users on your system should not know the password to LOCK_CLI. Changing the LOCK_CLI password is described later in Chapter 14, "Maintaining System Security."

To execute LOCK_CLI, type X :LOCK_CLI and press NEW LINE. To run it as a matter of course, edit the UP.CLI macro. In the UP.CLI macro, change the line that says EXECUTE CLI to EXECUTE LOCK_CLI. Then insert SUPERUSER ON before the EXECUTE line and insert SUPERUSER OFF after. The three lines should look like this:

```
SUPERUSER ON
EXECUTE LOCK_CLI
SUPERUSER OFF
```

Now LOCK_CLI will run on the system console whenever you bring up the system by executing the UP.CLI macro.

LOCK_CLI retains the username OP, so it can issue commands to EXEC while locked.

Viewing and Comparing Files (BROWSE, DISPLAY, FILCOM, and SCOM)

The BROWSE and DISPLAY utilities can read any binary, ASCII, or EBCDIC file — on tape or disk — and display it or write it to a listing file. DISPLAY is useful when you want to read a tape file without loading it, when you want to see what an EBCDIC file looks like in ASCII, or when you want to see everything (including nonprinting characters) that a program wrote to a disk file. BROWSE is useful because it provides many features, including forward and backward scrolling, support of function keys (identified on the AOS/VS and AOS/VS II Menu–Based Utilities template), string searching, viewing of as many as three files at one time (each in its own window), on–line help, placemarks, and so on.

For example,

```
) X DISPLAY @MTB0:0 ↵

) X DISPLAY MYPROG.PR ↵

) X DISPLAY :SYSLOG ↵

) X DISPLAY/L=REAL_OUTPUT_FILE OUTPUT_FILE ↵

) X BROWSE :CON0_LOG ↵

) X BROWSE/FORWARD :CON0_LOG ↵
```

FILCOM and SCOM are file–comparison utilities. FILCOM compares binary files; SCOM compares ASCII text files. These utilities can be helpful when you want to see if two files differ, or how they differ. For example, if you have two revisions of a program and want to see if they differ, you'd use SCOM for the source file or FILCOM for the program file. For example,

```
) DIR NEW_PROGS ↵
) X FILCOM MPROG_1.PR MPROG_2.PR ↵

... (Listing to terminal) ...

) X SCOM/L=@LPT MPROG_1.SR MPROG_2.SR ↵

... (Listing to line printer) ...
)
```

The manual *Using The CLI (AOS/VS and AOS/VS II)* gives more information on BROWSE, DISPLAY, FILCOM and SCOM. Each of these utilitites is a Help topic. Anyone can get help with these topics by typing the name of the topic preceded by an asterisk. For example, to get help on BROWSE type

```
) HELPB *BROWSE ↵
```

Displaying the Process Environment (PED)

PED, the Process Environment Display utility, displays information about the active processes running on your system. It is designed for DASHER® D200–series, and D400–series CRT display terminals, but will also run on hardcopy terminals.

There are many different ways to invoke PED. The easiest way is simply to type

) PED ↵

which invokes the supplied macro PED.CLI, and gives you a display that looks like Figure 11–1.

```
 PID M USERNAME PROCESS  PROGRAM    ELAPS    CPU CPS BS IO/S FTAS PRI   WSS
   1 * PMGR     PMGR     PMGR       21/36   3:01   1        0    0   2   278
   9 # OP       DCS      DCS         2/09   5:48 389        0    0   2   843
  30 # CEO_MGR  CEO_FSA  CEO_FSA     9/23  13.80  77        0    7   2   202
  57 # OP       SMAP     SMAP        2/10  35.95   7 B      0    0   1   235
  59 # OP       XTS      XTS         2/09  37.49   1        0    0   2   480
  77 # OP       MTA      MTA         2/08  16.33   3 B      0    0   2   387
  91   PMAC     00091    SED        20:08   3.04   5        0    0   2   194
  94   JONATHAN 00094    PED        30.00   0.32   8        0    0   2   171
  97   LURIE    RCP_96   CC         59.00  13.52 162       10   11   2  1140
 270   SHAWN    CEO_CP_5 CEO_CP      1/58   4.03   0 B      0    0   2   390
 274   MALC     CEO_CP_6 CEO_CP     19/54  24.57   0 B      0    0   2   526
 283   CLI_32   CEO_CP_6 CEO_CP      4:58   4.59   8 B      4    3   2   384

PED Rev 3.00.00.00   Wednesday  4-Aug-93 9:36:05 AM DOC4
```

Figure 11–1 Sample PED Display

By default, PED displays only active processes. If you execute PED, either through the PED.CLI macro, or by typing XEQ PED, a default set of switches appears as the PED initial screen.

Table 11–3 explains the various columns of the PED display.

Table 11–3 The PED Display, Column by Column

Column	Shows
PID	The process ID.
M	Indicates if Superuser or Superprocess (mode) is on (for user processes).
USERNAME	The username; first eight characters only.
PROCESS	The system name for the program process. You can assign this name by including /NAME= in the PROCESS command. Only the first eight characters are displayed.
PROGRAM	The program filename (first 10 characters).
ELAPS	The amount of time the process has existed.
CPU	The amount of CPU time used.
CPS	Rate of CPU time used (in milliseconds per second).
BS	Blocked, swapped to disk, neither (2 spaces), or both.
IO/S	The number of 512–byte disk blocks transferred per second via the data channel.
FTAS	The number of pages that had to be read from disk because they weren't in virtual memory (page faults). Page faults are an important indicator of program efficiency.
PRI	Shows process priority.
WSS	The number of pages in the process working set of pages in main memory.

By default, PED displays only those processes that have changed since the last screen update. The default screen cycle period is 10 seconds, but you can change this with runtime commands or the /CYCLE= switch; the minimum is 1 second. You can also update the display by striking any key other than a PED command key. The /ALL switch tells PED to display all processes, but in many systems all the PIDs won't fit on a screen.

The default PED display is useful in many situations. You can get a different display by using switches when you execute PED. While in a PED session, you can dynamically change the PED screens display by typing M (menu) and selecting different statistics for display. You can also issue runtime commands to PED.

PED is a HELP topic; any user can get help by typing HELP *PED and pressing NEW LINE.

PED Switches

There are two types of PED switches: those that select process statistics for display, and those that set PED parameters. If you select statistics (for example, /FTP), PED will display only those you select. The parameter group includes /ALL, /CYCLE, /LISTFILE, and /MINPID. If you use parameter switches and no statistics switches, PED will display all statistics shown above.

You can abbreviate all switches shown in Table 11–4. PED ignores ambiguous abbreviations and nonexistent switches.

The switch order is unimportant: PED always displays columns in a specific order, regardless of your switch sequence.

Table 11–4 PED Switches

Switch	What It Displays or Does
/ALL	All PIDs; the default is active PIDs. On a CRT, active PIDs are bright; others are dim.
/BATCH	Runs PED in batch mode, freeing up your terminal. PED ignores runtime commands when you use this switch. Use /BATCH with /LISTFILE. For example,) PROC/DEF/PED/BATCH/CYCLE=1/SNAP=10/LIST ↵
/BS	Blocked and swapped status.
/CLASSID	Class ID for each process; column header is CID.
/CLASSNAM	Class name for each process. Class names are created via the optional Class Assignment and Scheduling Package (CLASP).
/CPU	CPU time used (total).
/CPUS	Milliseconds of CPU used per second. Each unit is 0.1% of the total CPU time. For example, 500 means 50% of the total CPU time. The column header is CPS.
/CYCLE=*n*	Sets a cycle time of *n* seconds between updates.
/ELAPSED	Process elapsed time; column header is ELAPS.
/FATHER	Father process ID, under head FP.

(continued)

Table 11–4 PED Switches

Switch	What It Displays or Does
/FLAG*n*	Flag word *n*, where n is 1 through 5 (the flag words contain process parameters, described under ?PSTAT in the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q*).
/FTA	Page faults, sum of logical and physical.
/FTAS	Page faults, sum of logical and physical, per second.
/FTL	Page faults, logical.
/FTLS	Page faults, logical, per second.
/FTP	Page faults, physical.
/FTPS	Page faults, physical, per second.
/IO	I/O blocks (number of 512–byte blocks transferred to disk or tape).
/IOS	I/O blocks per second; column header is IO/S.
/IREC	Number of process tasks that are waiting for an IPC message (are waiting on an ?IREC system call). The column header is IR.
/LISTFILE*[=path]*	Sends PED output to file named in *path;* if you omit *=path*, it sends the output to the default list file PED.LIST file. PED uses /ALL for the first write to the file; then it writes the specified or default display at each cycle period.
/MAXPID=*n*	This displays only those PIDs equal to or less than *n*. Runtime commands can change this.
/MINPID=*n*	This displays only those PIDs equal to or greater than *n*. Runtime commands can change this.
/MPROCESSOR	Displays whether mother job processor only can run the process. The column header is MP. Y means yes.

(continued)

Table 11–4 PED Switches

Switch	What It Displays or Does
/PAGESECONDS	Page–seconds: the integral of (pages–used * CPU–secs–used). The column head is PGSEC.
/PID	Process ID.
/PIDSIZE	Process PID–size type: A, B, or C. The column header is PSZ.
/PLOCALITY	Process program locality. The column header is PL.
/PNQ	Priority enqueue factor (indicates the absolute priority of a process, regardless of its type). See the section "Processes and CPU Time" in Chapter 13 for an explanation of swappable, resident, and preemptible processes. to other processes of the same type).
/PRIORITY	Process priority, under column head PRI.
/PRIVBITS	Process privilege word (?PPRV, in the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q);* column header is PRIVS.
/PROCESS	Process name.
/PROGRAM	Program name.
/PRTYPE	Process type. PED displays the process type of each PID within its range of display. The display abbreviations are: SWP for swappable RES for resident, and PRE for pre–emptible.
/PSW	Process status word (?PSTAT, in the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q*).
/RANGE=*n*	This displays only those PIDs between MINPID+*n* or MAXPID, whichever is less.
/SH	Shared pages in rings 3–7 (cumulative).
/SH*n*	Shared pages in ring *n*. *n* can be an integer 3 through 7. The default displays ring 7 only.

(continued)

Table 11–4 PED Switches

Switch	What It Displays or Does
/SNAP=*n*	Has PED collect data for *n* times. See /BATCH for how to use this switch.
/SUBSLICES	Number of subslices in time slice; the column header is SL.
/SUPERMODE	Superprocess and Superuser status in CLI prompt format. + means Superprocess, * means Superuser, # means both. The column header is M.
/SUPERPRIVILEGES	Superprocess and Superuser privileges in CLI prompt format; the column header is P.
/SYSMGR	System Manager privilege. The column header is S. Y means yes.
/TSE	Time slice exponent; the column header is E. See /SUBSLICES.
/ULOCALITY	Process current user locality. The column header is UL.
/US	Unshared pages in rings 3–7 (cumulative).
/US*n*	Unshared pages in ring *n*, range 3 through 7. The default displays ring 7 only.
/USERNAME	Username.
WSS	Working set size, current.
/WSSMAX	Working set size, maximum, as given (or defaulted) in user profile.
/WSSMIN	Working set size, minimum, as given (or defaulted) in user profile.

(concluded)

Invoking PED with no arguments is equivalent to typing

) PED/CYCLE=10/PID/SUPERMODE/USER/PROCESS/PROGRAM/ELAPS& ↵

&)/CPU/CPUS/BS/IO//FTAS/PRIORITY/WSS ↵

PED Menu

The PED menu allows you to select different statistics for display. Pressing M during a PED session displays the PED menu. Figure 11–2 shows the PED Menu.

```
Cursor positioning chars:     U - up    D - down   L - left   R - right
Switch selection   chars:     Y - yes   N - no     S - default switches
Misc:         SPACE - refresh screen   BREAK/ESC - exit this menu

 1  BS          Y    15  FTL             29  PRIVbits         43  sysMGR
 2  Class ID        16  FTLS            30  PROCESS     Y    44  tsE
 3  CLASSNAM        17  FTP             31  PROGRAM     Y    45  ULocal
 4  CPU         Y   18  FTPS            32  PRTYPE           46  US
 5  CPUS        Y   19  IO              33  PSW              47  US3
 6  ELAPSed     Y   20  IOS         Y   34  SH               48  US4
 7  Father P        21  IRec            35  SH3              49  US5
 8  FLAG 1          22  MProcess        36  SH4              50  US6
 9  FLAG 2          23  PaGeSEC         37  SH5              51  US7
10  FLAG 3          24  PID         Y   38  SH6              52  USERNAME    Y
11  FLAG 4          25  PID SiZe        39  SH7              53  WSS         Y
12  FLAG 5          26  PLocal          40  SubsLice         54  WSsMAX
13  FTA             27  PNQ             41  suprMode    Y    55  WSsMIN
14  FTAS        Y   28  PRIority    Y   42  suprPrivg

Width is  76
```

Figure 11–2 The PED Menu

When you first see the PED Menu, the default PED switches (or those switches you selected the last time you executed PED), have a Y displayed. At the bottom of the menu the total width of the PED columns, 76, appears. The first line of the menu shows the keys to press to move the cursor around the menu. You can also use the cursor control keys. Move the cursor to the item you want to change or select. Then type N if you don't want to display an item already displayed, or Y to select an item.

At all times, the menu keeps a running tally of the total width. Since PED cannot display more than 80 characters, the PED menu won't let you leave the menu if the total width is more than 80 characters. Delete items until the total falls to 80 or less. Press BREAK/ESC to exit from the menu and return to the PED screen, which will show the items you just selected.

After you have experimented with the PED Menu, you may want to write a macro selecting the switches corresponding to the column headings you're interested in. This macro will save you time in the future.

PED Commands

You can type any of the commands in Table 11–5 while PED is running.

Table 11–5 PED Commands

Command	What It Does
A	Changes /ALL to active PIDs; then active to /ALL, and so on.
B	Bye; returns to the CLI.
E	Exit; returns to the CLI.
H	Help; gives help.
M	Displays PED Menu.
Q	Quit; returns to the CLI.
R	Refreshes the screen; useful after the screen has been changed by a message sent by another user.
CTRL–S	Freezes display; works as with any program.
CTRL–Q	Unfreezes display; works as with any program.
^ (Shift–6)	Increases minimum PID number (MINPID) by 5.
V (V key)	Decreases MINPID by 5.
]	Increases PID n by 20, as n was specified with /RANGE=n switch.
[	Decreases PID n by 20, as n was specified with /RANGE=n switch.
}	Scrolls screen up 5 PIDs.
{	Scrolls screen down 5 PIDs.
<	Decreases PED cycle time by 1 second.
=	Resets PED cycle time to 10 seconds.
>	Increases PED screen cycle time by 1 second.

The B, E, or Q command leaves PED. Unless you use the /LISTFILE switch, avoid exiting with CTRL–C, CTRL–B (or you will need to type CHAR/RESET and press NEW LINE to restore your terminals screen characteristics).

PED Abbreviations

For compact display, PED uses the following abbreviations:

dd–hh	days–hours.	For example, 4–03 means 4 days and 3 hours.
hh/mm	hours:minutes.	For example, 3/27 means 3 hours, 27 minutes.
mm:ss	minutes:seconds.	For example, 27:03 means 27 minutes, 3 seconds.
ss.nn	seconds.fractional seconds.	For example, 3.50 means 3 and 1/2 seconds.

If a PID has run for more than 99 days and 23 hours (99–23), PED displays its time as *****.

If a number will not fit in its field, PED divides it by a thousand (integer division) and appends K to it. If it won't fit when divided by a thousand, PED divides it by a million and appends M to it.

For example, in a five–character field,

65899	means 65,899.
199K	means between 199,000 and 200,000.
2100K	means between 2,100,000 and 2,100,999.
12M	means between 12,000,000 and 13,000,000.

Many of PED's features are designed to help programmers streamline and speed up their programs. In its general form, PED can give anyone either an overview or very specific information. You'll find it increasingly useful as the number and complexity of the processes on your system grow.

For general system operations, you may tend to use the ?.CLI macro and RUNTIME command more than PED. For system and program analysis though, PED is an invaluable tool.

PED Examples

```
) PED ↵
... (Display) ...

B


) PED/CYCLE=60/L=PED_LISTING_FILE ↵

CTRL–C CTRL–B
*ABORT*
*CONSOLE INTERRUPT*


) TYPE PED_LISTING_FILE ↵

...


) PED/ALL/MINPID=5 ↵

... (Display) ...

V
...
V
...
B
```

Logging Operating System Events (ERROR_LOG, SYSLOG, Superuser Logging, and CON0_LOG)

Through SYSLOG, the system logging function, the operating system always records hardware errors in the system error log, :ERROR_LOG. SYSLOG optionally lets you record user event information in the file of the same name, :SYSLOG. With CLI32, SYSLOG lets you use Superuser logging to log events, caused by a superuser, that would normally be logged only if you chose full–detail logging. With CLI32, SYSLOG also lets you log to a disk file the I/O that appears on the system console to a third log file, :CON0_LOG.

SYSLOG offers a protection feature, which prevents anyone from making changes to system logging while the system is up. See the /PROTECT switch, described in the section "Controlling Error, System, Superuser, and CON0 Logging," for more details.

Device error information, like entries for hard disk errors or ERCC memory errors, is recorded in the error log file (pathname :ERROR_LOG). The error log can be very useful to Data General diagnostic personnel, since it continuously records hardware errors. Error logging is always on (you cannot turn it off) to make sure error records will be available if needed. The error log includes information on hardware errors and operating system errors like hangs or panics.

User account–related information (gathered by the operating system and EXEC) and XODIAC network information (gathered by networking software and the operating system) are optionally recorded in the system log file (pathname :SYSLOG). If you choose full–detail logging, you can also record file access information, gathered by the operating system and EXEC, to help you monitor the level of system security.

Users and programs can write additional information to or mask certain events from the system log file:

- with the CLI command LOGEVENT. Any superuser can write messages into the log file with the LOGEVENT command;
- with the assembly language system call ?LOGEV (Superuser mode required), described in the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A Through ?Q*;
- with the assembly language system call ?SYLOG (System Manager privilege must be turned on), you can manipulate the system log just as you can via CLI commands. You can also exclude specific user events for all users by including a general exclusion bit map, or exclude specific events caused by a superuser by including a superuser exclusion bit map. ?SYLOG is described in the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R Through ?Z.*

:SYSLOG and :ERROR_LOG are not directly readable. You can use the REPORT utility, described later in this chapter in the section "Reporting Operating System Events (REPORT)," to read these files or produce readable disk file reports from them. You can also write a program to create a report. See an appendix to the manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R Through ?Z*, which documents SYSLOG record formats.

The file CON0_LOG is readable (it has a default ACL of OP,R): user OP can use the CLI TYPE command to display it or the DISPLAY or BROWSE utility, described earlier in this chapter, to view it or to search for text strings in it. (If you want others to be able to view the file, change its ACL to +,R in the system's UP.CLI macro.)

NOTE: To view the CON0 log file, stop logging in order to flush the system buffers. This will ensure that the log file is up to date. If you don't stop logging, log information may not be up to date and may appear truncated until the next buffer is flushed. Then use the CLI TYPE command or the BROWSE or DISPLAY utility to view it. But do not view the log at the system console while CON0 logging is enabled or you will get into an infinite loop (since you will never get to the end of the file). If that happens, interrupt TYPE or BROWSE by typing CTRL–C, CTRL–A; interrupt DISPLAY by typing CTRL–C, CTRL–B.

Other Log Functions

In addition to these operating system logging functions, your ECLIPSE MV/Family computer, EXEC, and other software running on your system provide other logging functions. They include

- The System Control Program (SCP) log. On computers that have an SCP CPU, the SCP–OS always logs certain kinds of hardware errors. (All MV/Family computers except the 4000–series have an SCP CPU.) The SCP log file is on diskette for MV/8000 systems; it is retained in SCP memory for other systems. Some, but not all, material from the SCP log is written into file ERROR_LOG. Generally, the operating system error log will tell you what you need to know; you don't need the SCP log unless you're having problems.
- EXEC's log. EXEC has its own logging feature, described in Chapter 3. EXEC always writes user–relevant information to SYSLOG; so in terms of user logging, you don't need EXEC's log if you use SYSLOG.
- The Common Logger. This logging function is relevant if your system runs Data General's INFOS II file management system. (The Common Logger software ships with AOS/VS II.) The Common Logger records database–backup information on tape. It has no relation to SYSLOG.
- XODIAC network logging. This logging function is relevant if your system runs Data General's XODIAC network system. Via CONTROL commands, you can direct networking processes like X.25 and LMGR (Link manager) to log network events. Each XODIAC log file is separate from SYSLOG; it is usually in directory :NET. SYSLOG records some, but not all, of the network events recorded in the XODIAC log file(s).
- Communications product logging. This logging function is relevant if your system runs an IBM emulator like DG/SNA. These products have logging features similar to those of XODIAC; SYSLOG records some of their events.

Controlling Error, System, Superuser, and CON0 Logging

SYSLOG is the CLI command to control system logging. You can issue this command from an unlocked system console (PID 2). If you have the System Manager privilege, you can use this command to control system logging from a user terminal. (First issue the CLI command line PRIVILEGE SYSTEMMANAGER ON and press NEW LINE.)

NOTE: The operating system buffers SYSLOG, ERROR_LOG, and CON0_LOG events. Stopping logging explicitly or through renaming a log file flushes those events from the buffer and ensures that the log file is up to date.

The SYSLOG command has the following format:

SYSLOG*[/switches] [filename]*

where

SYSLOG	If you use SYSLOG alone, it tells you the status of system logging: OFF if system logging is off, or ON, if system logging is on. (Use the 32–bit CLI /VERBOSE switch to check SYSLOG logging options, Superuser logging, and CON0 logging.) You typically use SYSLOG with the switches listed next.
/CON0/START *[filename]*	(32–bit CLI). If you do not include a filename, the CLI starts logging in the current :CON0_LOG file. If there is no :CON0_LOG FILE, the system creates one. If you include a filename, this switch renames the current :CON_LOG to that filename, then starts logging in a new :CON0_LOG file. You cannot use other switches with /CON0/START. Be aware that file :CON0_LOG can grow quite rapidly. As with system logging with detail set to FULL, the system tries to continue logging at all costs: if the logical disk that holds the log file fills up, the system won't disable logging; it will panic. See the section "Disk Space Cautions" later in this chapter.
/CON0/STOP	(32–bit CLI). Stops CON0 logging. You cannot use other switches with /CON0/STOP. Suggestion: turn CON0 logging off when you want to run a system utility on the system console (and turn it on again when you are done) to avoid logging meaningless I/O.

/DETAIL=MINIMAL *[filename]*
/DETAIL=FULL *[filename]*

When used with the /START switch, sets the level of detail when you start logging. Used at any time with a filename, renames :SYSLOG to the filename you specify. If logging is already on, changes the level of detail.

Minimal–detail logging includes most user account information except for file access, including terminal connect time, CPU time, I/O blocks, pages printed, and other things. With detail set to minimal, if the logical disk that holds the log file fills up, logging will stop.

Full–detail logging includes all of the account information above, plus user access (audit trail) information. This setting records events that can relate to system security, as follows:

- all file access attempts (successful or unsuccessful);
- all file ACL changes;
- all file renames or deletes;
- all process creations (with process privileges), process terminations, and ring loads (?RINGLD system call);
- all attempts to turn Superuser and Superprocess on or off.
- an error code, if an error occurred on the event. If no error occurred, the user's call succeeded.

Be aware that all this information will make file :SYSLOG grow quite rapidly. With detail set to FULL, the system tries to continue logging at all costs: if the logical disk that holds the log file fills up, the system won't disable logging; it will panic. See the section "Disk Space Cautions" later in this chapter.

filename

If you specify a filename without switches, renames :SYSLOG to this name. With system logging already started, renames :SYSLOG to this name and restarts :SYSLOG with the level of detail that was previously in effect.

/NOSOFTTAPEERRORS

Tells the system not to record soft tape errors in the error log file or display them on the system console. More than three soft tape errors, however, will be logged. A soft tape error is an I/O inconsistency that disappeared in fewer than 13 retries. (If it persisted for more than 13 retries, it would be a hard error.) By default, soft tape errors are logged in the error log file. However, certain brands of magnetic tapes can produce many soft error messages, causing the log file to grow rapidly and slowing response. You can use this switch, or /SOFTTAPEERRORS, without having to stop and then restart logging.

/PROTECT*[/CON0]*

CLI32 only. Protects the file :SYSLOG or the file :CON0_LOG while the system is running. Protection ends when the system is brought down. Protection prevents anyone, including a Superuser or a process with the System Manager privilege, from

- deleting or modifying :SYSLOG or :CON0_LOG
- renaming :SYSLOG or :CON0_LOG
- starting or stopping logging
- changing the level of detail
- setting an exclusion or superuser bitmap

Enabling protection is logged to :SYSLOG (provided, of course, that system logging is enabled).

Protection does not prevent viewing logging status, viewing CON0_LOG or getting a report of logging events, and protection has no effect on :ERROR_LOG.You cannot use /PROTECT or /PROTECT/CON0 with any other switches.

WARNING: Because protection prevents stopping SYSLOG except through shutting down the system, be sure that there is sufficient space for the SYSLOG to grow. Otherwise, you will not be able to prevent a growing SYSLOG from causing the system to panic. Also, make sure you have chosen logging options correctly before using /PROTECT.

/RENAMEERROR	Renames :ERROR_LOG to the specified filename, and then continues error logging in a new :ERROR_LOG file. The system starts recording hardware errors in a new, fresh ERROR_LOG. This is useful when you want to dump and/or delete an error log file.
/SOFTTAPEERRORS	Records soft tape errors in :ERROR_LOG. This switch is useful if you want to record soft errors and the current VSGEN system was generated with soft tape error reporting suppressed. (See /NOSOFTTAPEERRORS above for more information.)
/START	If you do not include a filename, tells the system to log minimal detail to the current system log file, :SYSLOG. If you include a filename, this switch renames the current :SYSLOG to that filename, then starts logging in a new :SYSLOG file.
/STOP	Stops recording in the system log file and stops Superuser logging if on. (Error logging continues.) Shutting down the operating system automatically stops recording in the system log file, so you don't need to use this switch before shutting down.
/SUPERUSER/START	(32–bit CLI). Starts Superuser logging in the current :SYSLOG file. You must start system logging before starting Superuser logging. Therefore, you cannot use other switches with these switches.
/SUPERUSER/STOP	(32–bit CLI). Stops Superuser logging. You cannot use other switches with these switches.
/VERBOSE	(32–bit CLI). Displays status of system logging, Superuser logging, and CON0 logging.

Log File Pointers and Suggestions

Logging is useful for the following reasons:

- It can track the amount of system resources used by each user. This is useful for billing or other user monitoring situations in a time–sharing system.
- It can record file accesses and use of special privileges and processes — events that may affect system security. Such a record can be critically important on a system used to maintain sensitive or classified databases and programs.
- It records hardware errors. Hardware errors are logged automatically.
- With detailed logging, you can lose information if you stop or start logging while the system is active.

If you are concerned with security, do not stop or start the system log or rename the system log while the multiuser environment is active. If you do, you will interfere with SYSLOG's ability to get information you need. For example, any processes that are running when logging starts will have only PID number (not username) data available for reports. Also, any files that are open when logging starts will have only PID and channel number (not pathname) data available.

For detailed logging, we suggest that you start the system log immediately after the master CLI comes up, before EXEC or any other process. (The UP.CLI macro is a good place from which to start logging.)

If logging is on and you have not protected :SYSLOG, you can rename file :SYSLOG and start a new one. This is useful when you want to dump and delete the old log. To rename :SYSLOG, use the form SYSLOG filename. Logging resumes at the same level of detail that was previously in effect.

If you can, however, we recommend shutting down every process except the master CLI (PID 2) before renaming :SYSLOG. Shutting down processes will limit the processes recorded without a username to PID 1 and 2 (PMGR and master CLI), and these processes' usernames (PMGR and OP) are easy to identify in reports.

To be most useful, logging should be consistent, and log records for at least the last 12 months should be available. Because system logs, especially those with detail set to full, grow so rapidly, you need a procedure for naming, archiving, and deleting them.

The current log pathnames are always :SYSLOG and :ERROR_LOG. (AOS/VS II also lets you log to file :CON0_LOG.) One way to identify old logs is to rename the current one to a name that indicates the date it was started. You might do this every week or more often for the SYSLOG file, which grows rapidly, and every month or so for the error log file, which grows slowly. This way, the filename will show the end of the logging period covered by the file. For the date, you could use the form

ddmonyy

where ddmonyy is the date on which the preceding log file was started, for example, 30–JUN–93. This filename identifies each log clearly and uniquely. For more clarity, you can add the suffix .LOG or LOG.ERROR to the filename.

You can rename an old log file (starting a new log file) at any time. Commands to rename have the form

SYSLOG/START *[other–switches]* name–for–old–logfile

SYSLOG/RENAMEERROR name–for–old–errorfile

For example, a sequence might go

AOS/VS II CLI Rev date time	Bring up the operating system.
) DATE↵ *27–Sep–93*	Check the date. It's September 27th.
) SYSLOG/START 27SEP93.LOG ↵	Rename old system log file to previous date and start new system log file (SYSLOG).
) SYSLOG/RENAMEERR 27SEP93.LOG.ERROR ↵	Rename old error log file to previous date and start new error log file (ERROR_LOG).
) SUPERUSER ON ↵	Turn on the Superuser privilege.
Su) LOGEVENT Running new rev ↵	Log an event in system log.
Su) UP ↵	Bring up the multiuser environment.
... (System runs through the day) ...	
Su) DOWN ↵	Bring down the multiuser environment.
Su) BYE ↵ *SYSTEM SHUTDOWN*	
SCP–CLI> BOOT 24 ↵ ... *AOS/VS II CLI REV ....*	Boot from disk.
) DATE ↵ *01–Oct–93*	Check the date.
) SYSLOG/START 01OCT.LOG ↵	Rename the system log and start a new one.
) SYSLOG/RENAMEERROR 01OCT.LOG.ERROR ↵	Rename the error log and start a new one.
) UP ↵	Bring the system up, once again.

There are several different approaches to beginning a new log. An easy way is to start fresh logs each time a system is brought up by commands like those shown above. You can't always start fresh logs at startup though, since bringing up the system twice on the same day would mean renaming logs to existing names. Ideally, you'd rename the logs only if they weren't started today. A macro to do this is shown in Figure 11–3. This macro starts the system log, and renames both logs to filenames that show the date they were started if logs weren't renamed today. If disk space is an issue — and with full–detail logging it usually is — you might also use the CHECK_SPACE macro that is designed to warn you when disk space is low. See the section "Disk Space Cautions," later in this chapter.

Text of The SYSLOG_UP.CLI Macro

```
[!equal,%0/%,/H]
        write  This macro starts the system log with a SYSLOG command
        write  that renames the old system and error log files to the
        write  date they were started. Then it moves the old system log
        write  and error files into directory :LOGS. If the old :SYSLOG
        write  was started today, the macro starts :SYSLOG and doesn't
        write  rename the error log — appending new information to the
        write  existing :SYSLOG and :ERRORLOG instead of renaming and
        write  moving them. You could name this macro :SYSLOG_UP.CLI and
        write  call it from the UP macro.
        write  Before the macro will work, someone must:
        write  ,, 1. Create it in the root directory.
        write  ,, 2. Create macro MAKE_DATE.CLI in the root directory.
        write  ,, 3. Create directory :LOGS  — e.g., via CREATE/DIR :LOGS
        write  After these steps have been done, the macro will do log
        write  housekeeping simply via the command ,, SYSLOG_UP. It takes
        write  no arguments but the /H switch produces this Help message.
[!else]
                push
                superuser on
                dir  :
                string [MAKE_DATE [!explode [!date]]].LOG
                [!nequal,([!filenames LAST_LOG_DATE]),()]
                comment  For security, use /DETAIL=FULL. Add this switch to
                comment  the two syslog command lines.
                        push
                        string [LAST_LOG_DATE]
                        comment  If today's date differs from the date stored in
                        comment  file LAST_LOG_DATE, rename the log files and
                        comment  move them.
```

Figure 11–3 Example SYSLOG_UP and Companion Macros (continued)

```
                    [!nequal,[!string/p],[!string]]
                            syslog/start/1=warning/2=warning  [!string]
                            syslog/renameerror  [!string].ERROR
                            acl  [!string]  OP,OWARE +,RE
                            write Moving log files [!string] and
                            write [!string].ERROR to :LOGS
                            move   LOGS  [!string]  [!string].ERROR
                            delete  [!string]  [!string].ERROR
                    [!else]
                            comment SYSLOG was started today; start logging.
                            syslog/start
                    [!end]
                    pop
                    [!else]
                            write File ,, LAST_LOG_DATE ,, doesn't exist. I will
                            write create it. Please rerun this macro by typing %0%
                            write after  it stops.
                    [!end]
comment Update — or create — the LAST_LOG_DATE file.
delete/2=ignore  LAST_LOG_DATE
string/l=LAST_LOG_DATE
acl  LAST_LOG_DATE   +,R
pop
[!end]
Text of MAKE_DATE.CLI

%1%%2%%4%%5%%6%%8%%9%&
```

Figure 11–3 Example SYSLOG_UP and Companion Macros (concluded)

Assume you run the SYSLOG_UP macro shown in Figure 11–3 on November 27th (after following setup instructions in the macro). The log pathnames will remain

:SYSLOG (For system log)
:ERROR_LOG (For error log)

Then, on November 28th, you run the macro again. Since the dates differ, the macro renames the logs and moves them to directory :LOGS. The current log filenames remain (as always) SYSLOG and ERROR_LOG. In this case, the logs would have the pathnames

:LOGS:28NOV91.LOG (system log)
:LOGS:28NOV91.LOG.ERROR (error log)

Later on, at reasonable intervals, you can dump and delete log files in directory :LOGS. If desired, before deleting them, you can create reports (using the REPORT program); then print and delete the report files. The REPORT step depends on whether standard reports (which can be done as a matter of routine) will satisfy your needs.

In any case, if you need a report on a given period, you can always reload the dump file that holds the desired logs and then run REPORT on these files.

You can write your own program to manipulate SYSLOG records. For more information, read about the SYSLOG record format in an appendix to the manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R Through ?Z.*

Disk Space Cautions

As mentioned earlier, log files can grow to consume a large amount of disk space. Large log file are most critical with full–detail logging and with CON0 logging, especially if you run XODIAC networking on your system. If you start logging with detail set to full or run CON0 logging, the operating system will panic if either log file grows to the point where it runs out of disk space. One way to avoid this is to write a program that filters the information that actually gets logged. See the system call ?SYLOG in the manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R through ?Z.*

The next two sections suggest another way to avoid running out of disk space. The first section shows how to put the system log file in its own directory, ideally on its own logical disk. The second section shows how to use the CHECK_SPACE macro that monitors disk space and warns you when space becomes crucial.

Whatever the degree of logging (minimal or full), and even if you devote a whole logical disk to the log file, you need to monitor disk space carefully. Periods of intense activity (like heavy interactive use and backup) make the log file grow rapidly.

The space–warning threshold should leave space for report generation. Then if disk space reaches the warning point, space will remain for a report. After the warning, the operator can rename the log (starting a new one), generate a report from the old log; then print the report, and dump and delete the log. (The SYSLOG_UP macro shows a way to rename the old log and start a new one automatically. The CHECK_SPACE macro can warn of low disk space.)

Using a Specific Log Directory

With system logging, especially with detailed logging, and with CON0 logging, you should put the log files in their own directory, ideally a separate logical disk. Creating a separate directory for the log files will minimize file creations and deletions in the root directory, as well as minimize file fragmentation and I/O to the system disk.

Logging occurs to files :SYSLOG and :CON0_LOG on the master logical disk, but :SYSLOG and :CON0_LOG can be link files to log files in another directory, in the master logical disk or another logical disk. For example, suppose you have an LDU called LOGS (the LDU filename you gave the LDU when you created it). In the root of the master logical disk, you create a link to SYSLOG on the log LDU:

Su) CREATE/LINK :SYSLOG :LOGS:SYSLOG ↵

Your UP.CLI macro can initialize the logical disk. After initializing, the UP.CLI macro must conditionally create a file of type LOG on the logical disk; then it should assign a null ACL to file SYSLOG. (If the SYSLOG and CON0 files are not type LOG, the system will conclude they are not real log files. It will then delete the link and create a SYSLOG file or a CON0_LOG in the root. Creating a file of type LOG is easy; use the CREATE command with the /TYPE= switch.)

A sample command sequence for the UP.CLI macro (using the LDUname LOGS) is

```
...
INITIALIZE disk–unit
[!EQUAL,[!PATHNAME LOGS:SYSLOG],]
CREATE/TYPE=LOG LOGS:SYSLOG
ACL/K LOGS:SYSLOG
[!END]
SYSLOG_UP ...
...
```

Then logging will occur to file SYSLOG in the LOGS directory. Giving the log file a null ACL prevents access to it by unauthorized users.

The log directory should have ample disk space: at least four times the space you expect the log file ever to consume. This space is a safety factor for log file growth and allows you to generate reports in the log directory. Also you can load old log files here and create reports from them — to check user or file access history.

(By the way, it's not possible or necessary to put the error log in its own directory. It's not possible because the error file, ERROR_LOG, is started before the master CLI starts, which makes it impossible to initialize an LDU for it. It's not necessary to put the error log in its own directory because error logs don't take up much disk space. If you want to, you can create a link named :ERROR_LOG to another directory on the master logical disk.)

If you decide on a nondefault logging directory, and want to use the SYSLOG_UP macro shown in Figure 11–3, you'll have to edit the macro to fit your own arrangement. For example, edit the macro if you want to specify full–detail logging or if your logging directory has a name other than LOGS.

Another macro you might want to include in your UP.CLI macro — after starting SYSLOG but before executing LOCK.CLI — is the space–checking macro shown in the next section.

Checking Disk Space

The macro shown in Figure 11–4 is designed to warn you when disk space is low. The macro runs, checks space, pauses for 30 minutes, runs again, and continues this cycle until you terminate it. If the amount of disk space remaining in the log directory or its parent directory shrinks to less than 5,000 blocks (about 2,500,000 bytes), the macro sends warning messages to the system console, PID 2.

The name of this macro is CHECK_SPACE.CLI. You may want to run it as a matter of course if you use detailed logging. The CLI that runs it needs Superuser privilege (to find SYSLOG's pathname). You must change the 2 in line 1 of the macro to a 1. Both numbers must be 1 in order for the macro to execute. If they do not match, you will receive a warning message telling you to change the 2 to a 1.

The best way to use the macro is to create a subordinate CLI from PID 2, and have this CLI run the macro. You could create the subordinate CLI and start the macro running as part of the UP.CLI macro, before your UP macro executes the lockable CLI. Use a command line like this:

```
PROCESS/NAME=CHECK_SPACE/INPUT=@NULL/OUTPUT=@NULL/DEF&
:CLI CHECK_SPACE.CLI
```

The new, space–checking CLI will run the macro every 30 minutes (or a different period if you want to change the PAUSE line near the macros end). You can check or terminate the subordinate CLI by PID number or by the process name CHECK_SPACE from the master CLI or any superprocess, for example

```
) RUNTIME CHECK_SPACE ↵
ELAPSED: 22:10 ...
) SUPERPROCESS ON ↵
Sp) TERMINATE CHECK_SPACE ↵
Sp)
```

If you decide to have a CLI run CHECK_SPACE from your UP macro, you might also choose to terminate this CLI from your DOWN macro.

The macro expects one of the following directory structures:

```
:SYSLOG–logfile

:control–point–directory : SYSLOG–logfile
:logical disk–directory : control–point–directory : SYSLOG–logfile
```

It will run with some other arrangements, but we suggest one of those above.

Figure 11–4 shows the text of the macro CHEK_SPACE.CLI.

```
[!EQUAL,1,2]
[!equal,comment,] This macro checks the amount of space left for log file SYSLOG. It
requires the log file to be in a control point directory, on the master logical disk or other
logical disk, in one of the following directory arrangements.
:              :                :

log file   CPD (or LDU) directory   LDU directory

log file          CPD directory

log file


To run this macro, a CLI process needs Superuser privilege. The CLI also needs E
(execute) access to the log directory and its parent directory (if any).

On startup, if this is the first check, set up needed files to be used while the macro
runs. Use VAR9 as an indicator: a VAR9 value  of 99999 means the CHECK_SPACE
files are ready. [!end]

[!nequal [!var9],99999]
permanence/2=ignore :UDD:[!username]:??CHECK_DIR   off
delete/2=ignore  :UDD:[!username]:??CHECK_DIR
create/max=100   :UDD:[!username]:??CHECK_DIR
permanence   :UDD:[!username]:??CHECK_DIR   on
write/l=:UDD:[!user]:??CHECK_DIR:ARG2.CLI
write [!ascii 045]2[!ascii 045][!ascii 046]
write/l=:UDD:[!user]:??CHECK_DIR:ARG6.CLI
write [!ascii 045]6[!ascii 045][!ascii 046]
[!equal,comment,] Set VAR0 to danger threshold of disk space. Original
figure is 5,000 blocks for a check each 30 minutes. [!end]

var0  5000
[!equal,comment,] Turn Superuser on to check the pathname and type of
the log file. Then turn it off. [!end]

superuser  on
pathname/2=abort/l=:udd:[!username]:??CHECK_DIR:?CHECK_SPACE_PATH.TMP
:SYSLOG files/nh/type/l=:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP
:SYSLOG superuser  off
string   [:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP]
string   [:UDD:[!username]:??CHECK_DIR:ARG2 [!string]]
[!equal,comment,]   The string has the SYSLOG file type, LNK or LOG. Use VAR8 val-
                                ue of 88888 to signify LNK. [!end]
```

Figure 11–4 CHECK_SPACE Macro to Check Disk Space Periodically (continued)

```
[!equal [!string],LNK]
VAR8 88888
[!end]
[!equal,comment,] Set variable VAR9 to show that setup is done. [!end]
var9 99999
[!end]

[!equal,comment,] Ready. See if SYSLOG is a link — VAR8 is 88888.
[!end]
[!nequal [!var8],88888]
[!equal,comment,] SYSLOG is a not a link — log files in the root.
[!end]
        delete/2=ignore :UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP
space/l=:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP :
string [:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP]

var1 [:UDD:[!username]:??CHECK_DIR:ARG6 [!string]]
[!equal,comment,] Set VAR2 high enough to keep it out of the way.
This lets us use the same check routine
regardless of log file directory. [!end]
        var2 4000000000
[!else]
        [!equal,comment,] SYSLOG is a link. Get the name of the real log
        file directory and parent directory. First get
        name of log file directory from pathname file. [!end]
string [:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE_PATH.TMP]
string [!directory [!string]]

delete/2=ignore :UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP
        space/2=abort/l=:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP &
        & [!string]
        string [:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP]

        [!equal,comment,] Extract space remaining from SPACE string. [!end]
        var1 [:UDD:[!username]:??CHECK_DIR:ARG6 [!string]]

        [!equal,comment,] Second, get name of log file parent directory
        — from pathname file. [!end]
        string [:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE_PATH.TMP]
        string [!edirectory [!edirectory [!string]]]
        delete/2=ignore :UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP
        space/2=abort/l=:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP &
        & [!string]
        string [:UDD:[!username]:??CHECK_DIR:?CHECK_SPACE.TMP]
        [!equal,comment,] Again, extract space remaining from SPACE
        string. [!end]
```

Figure 11–4 CHECK_SPACE Macro to Check Disk Space Periodically (continued)

```
var2  [:UDD:[!username]:??CHECK_DIR:ARG6 [!string]]
[!end]

[!equal,comment,] Disk space figures are in VAR1 and VAR2. Compare
                  them to the threshold. Notify — twice — if lower
                  figure is below threshold. [!end]
[!ult [!var1] [!var0]]
        send 2  [!ascii 214]
send 2 [!ascii 212] [!ascii 207]
send 2 [!ascii 216] *** Warning – Low Disk Space *** [!ascii 217]
send 2 ,,, On [!date] at [!time]
send 2 [!ascii 212]
        send 2 Space left in log directory is [!var1]. Threshold is &
        & [!var0].
        pause 5
        send 2  [!ascii 207] [!ascii 212]
        send 2 [!ascii 216] *** Warning – Low Disk Space *** [!ascii 217]
[!else]
        [!ult [!var2] [!var0]]
                send 2 [!ascii 212] [!ascii 207]
                send 2  [!ascii 214]
                send 2 [!ascii 216] *** Warning – Low Disk Space *** &
                &  [!ascii 217]
                 send 2 [!ascii 212]
                send 2 Space left in log directory PARENT directory is &
                &  [!var1].
                send 2 The threshold is [!var0].
                pause 5
                send 2 [!ascii 212]
                send 2 [!ascii 216] *** Warning – Low Disk Space *** &
                &  [!ascii 217]
        [!end]
[!end]
[!equal,comment,] Pause 30 minutes — 1800 seconds, and call this
                  macro again. [!end]
pause  1800
%0%
[!ELSE]
write This macro is nonexecutable. To make it executable you must
write use a text editor to change the 2 in line 1 to 1 — both numbers must be 1.
[!END]
```

Figure 11–4 CHECK_SPACE Macro to Check Disk Space Periodically (concluded)

Detail–log Panics

If a growing log file does force an operating system panic, follow these steps:

1. Run ESD and restart the operating system (but don't type UP yet). Before resuming normal processing, you must dump and delete the log file.

2. Check the log file size with the FILES/AS command. If the log file will fit on one tape, you can dump it and delete it without bringing EXEC up. (A 2,400–foot tape, at 1600 b/in, can hold a 39,000,000–byte file.) Dump the log to unlabeled tape (MTxn:0), delete it, and bring everything up.

3. If the log file won't fit on one tape, either start EXEC and use labeled tape or use DUMP_II. Use a manual PROCESS command, instead of the UP macro, to start EXEC. (The syntax is shown in your UP.CLI macro.) Then, via a labeled tape mount request, labeling tapes as necessary, dump the log file to labeled tape, and delete it. If you use DUMP_II, it can dump to multivolume unlabeled tapes. Bring everything up.

After this kind of panic, you might want to review your log file procedures, and perhaps generate reports and dump and delete log files more often.

User Writes and Reads with the System Log File

Any Superuser can send messages to the active SYSLOG file, under the general–purpose code, with the CLI command LOGEVENT. A user program with Superuser turned on can do the same thing with the ?LOGEV system call, if it uses event code 1065 (2051 octal). Either of these writes an event record into the log file (but neither returns an error if logging is not on). The maximum size for a log record is 1015 words.

SYSLOG files are not directly readable. But user programs can read these files if they use the proper record formats. Also users can read log files (including the active one) via the Display program. Most important, they can get reports on log files using the REPORT program. Qualified users can run REPORT on file SYSLOG even while logging is on.

As with all files, a user cannot read or write a log file without read or write access to it unless he or she has Superuser on. The access control list for file SYSLOG is null (only Superusers can read it). The access control list for ERROR_LOG and CON0_LOG is OP,R.

Reporting Operating System Events (REPORT)

The REPORT program interprets, sorts, and displays the records in system and error log files. REPORT is in :UTIL, filename REPORT.PR. It is available to any user process. It is a CLI Help topic, so anyone can get on–line help for the REPORT program by typing HELP *REPORT and pressing NEW LINE.

REPORT can write the logged information to any device or file you name. It can report specific types of information and can report on multiple log files.

If you run REPORT on a system log file, the report includes user information. If you run it on an error log file, the report includes device error information. The default report from a system log file is a summary of user information. The default report from an error log file includes hardware errors. You can specify both system and error log filenames to REPORT; if so, the report includes both user and device information. From any file, you can select specific information via switches.

If you omit switches, REPORT creates a default report. A default user report includes the following information for each user:

- Username (* means that this user has Superuser, Superprocess, System Manager, or Access devices privilege.),
- User tape or disk mount requests (if any),
- Console (terminal) connect time. The format is hours and minutes (HH:MM), but if the system log was started or stopped while the user was logged on, it is shown as 0:00,
- Time unit. Unit time is the total time charged for labeled tape use; that is, the total amount of time (in hours and minutes) a process had tapes mounted. This figure does not include unlabeled tape use,
- Number of pages printed,
- CPU time in hours, minutes, seconds and milliseconds,
- I/O blocks: the number 512–byte disk blocks read or written,
- Page–seconds. This relates memory usage and CPU time; it is memory pages used multiplied by the number of CPU seconds, and
- Number of processes created for this user during his or her connect time.

For example, the following commands set the search list to include the directory :LOGS, and execute the REPORT program to list the log for September 16th at the line printer.

Su) SEARCHLIST [!SEARCH],:LOGS ⤵

Su) XEQ REPORT/L=@LPT 16SEP93.LOG 16SEP93.LOG.ERROR ⤵

The REPORT program might produce reports that look like Figures 11–5 and 11–6.

USER SUMMARY FROM FILE(S): :16SEP93.LOG

USERNAME	CONNECT TIME CONSOLE	UNIT	PAGES PRINTED	CPU TIME	I/O BLOCKS	PAGE SECS	NUMBR PROCS
*$$DAVID	59:22	0:00	262140	1:14:08.108	257740	1648418	225
*$$LIZ	38:45	0:00	105	2:50:52.470	993652	2526872	478
$$PERFMGR	0:00	0:00	0	0:00:53.767	742	12576	15
*$$PHIL	60:28	0:00	4	1:41:51.445	186970	1827100	624
*$$RITA	14:04	0:00	94	0:20:46.126	3978	269465	40
$JUNE	27:58	0:00	589815	0:19:25.535	12045	414247	45
$MARG	16:01	0:00	0	0:11:08.773	4620	176030	26
$PAT	37:32	0:00	12	1:12:56.982	42700	2143416	63
$PRINTER	0:00	0:00	192	0:00:00.000	0	0	0
$STEPHEN	51:15	0:00	65549	1:42:32.378	44743	3562242	59
*$STEVEN	17:27	0:00	196605	0:22:09.464	18582	581633	65
CEO_MGR	0:00	0:00	0	0:10:00.353	8179	68111	15
DEMO	0:00	0:00	0	0:00:00.357	0	13	1
DEMO.MF	0:25	0:00	4	0:02:02.171	965	22957	9
*OP	167:19	0:00	5767089	2:31:47.277	149187	1942872	205
OPER	0:00	11:01	32	5:56:38.437	3376050	13702935	31
PMGR	0:00	0:00	0	0:38:44.045	1664	338863	1

NO DEVICE ERRORS RECORDED.
NO DEVICE SUMMARY WILL BE PRINTED.

Figure 11–5 Default Report from System Log (User Summary)

```
NO USER ERRORS RECORDED.
NO USER SUMMARY WILL BE PRINTED.

DEVICE SUMMARY FROM FILE(S): :16SEP93.LOG.ERROR,

DEVICE  UNIT        ERRORS
CODE          HARD  SOFT  TIMEOUT
22         0    0     31         1
```

Figure 11–6 A Default Report from an Error Log File

Running the REPORT Program

To run REPORT, use the command form

[QBATCH] XEQ REPORT *[switches] [logfile–pathname] [...]*

where

QBATCH runs REPORT in batch. Depending on the size of the log file(s), REPORT can take some time (say 15 minutes for a 2 megabyte log file) to create the report on line. You can run REPORT in batch to keep your terminal free.

switches select options as described described in the next section. If you omit switches, the report goes to the generic output file @OUTPUT. By default, @OUTPUT is your terminal.

NOTE: REPORT rquires that you type the entire name of each switch. If you abbreviate a switch name, REPORT will ignore the switch specification (displaying no error message). Be sure to type the full name of each switch. The maximum number of characters for all switches is 256.

logfile–pathname specifies the system or error log file from which you want the report. If you omit a logfile–pathname, REPORT will use all the log files in the working directory. (Log files are a special type of file, LOG. REPORT reports on all files of this type.)

For a report on the current log file (system or error), you will need to turn Superuser on, since the ACL for this log file is null. (If the ACL of SYSLOG isn't null, make it null when you start logging, since the log file often contains sensitive information.)

The operating system buffers both SYSLOG and ERROR_LOG events, and REPORT cannot report on events that the system has not flushed from those buffers. To flush the buffers and ensure that the system log file is up to date, use the following format

SYSLOG filename–for–old–syslog

For the error log file, use the format

SYSLOG/RENAMEERROR filename–for–old–errorlog

When it generates a report from multiple log files, REPORT processes the files in chronological order of creation time regardless of the order in which you specify them.

REPORT Switches

If you do not specify any switches, REPORT produces a default report on users or errors. You can request additional reports via one or more REPORT switches, described below.

REPORT can report only on events whose codes are recorded in a system or error file. These codes are recorded only when the operating system, or the emergency shutdown program (ESD), is running with logging on. There are some events that the operating system can't record, for the following reasons:

- The event forces the operating system to panic. Some events indicate an error condition so serious that the safest course is for the operating system to panic. If AOS/VS or AOS/VS II panics, ESD will write the panic information to the error log, but it will be written under the panic event code, not the specific error event code.
- The event occurs so often that logging overhead would degrade performance. If an event occurs very frequently in the course of normal operations, the operating system will not log it. This applies only to error conditions like a persistent memory parity error or disk error, not to security–related events. (The operating system logs everything if you specified a detail setting of FULL when you started logging.)
- The operating system is not running when the event occurs. For example, a BOOT command to the SCP cannot be logged, since the operating system isn't running when it occurs.
- If detailed logging was started after the multiuser environment was brought up, important information (like usernames) may not be available for reports. For example, REPORT can provide only the PID (not the username) of processes started before logging was started. Also, REPORT can give only the channel number and user PID (not the pathname) of files opened before logging was started.

  If you care enough about security to use detailed logging, you should start logging immediately after operating system startup (in the UP macro); don't stop until shutdown.
- The SCP is responsible for telling the operating system about the event, but doesn't do so (perhaps because the SCP, or its interface to the operating system, is having problems). Certain events can be detected only by the SCP. On one of these SCP–detected events, the SCP notifies the operating system of device code 45; then if logging is enabled, the operating system notes the event code in the error log file. If the operating system isn't notified, it cannot log the event.
- Implementation is planned for the future revisions of the operating system.

Under any of these circumstances, the pertinent event code will not be written to SYSLOG, and REPORT will not be able to report the event.

Several REPORT switches include the functionality of other switches. For example, the /C switch includes reports on all 30–odd CPU–related switches. So if you use the /C switch, you will see a report (usually with NO EVENTS RECORDED) for each CPU–related switch.

All REPORT switches and descriptions follow. The log event code, in decimal (followed by its octal value in parentheses), is part of the description.

Table 11–6 REPORT Switches

Switch	What It Does
/AFTER=*dd–mon–yy [:hh:mm:ss]*	Reports only the events that occurred on or after the specified date *dd–mon–yy* or time *hh:mm:ss*. This switch is relevant for both system and error log files. You can combine the /AFTER= and /BEFORE= switches to display any time period. For example, to view only those events occurring on August 4, 1993: *Su*) X REPORT/AFTER=4–AUG–93/BEFOR=4–AUG–93 & ↵ *Su&*)pathname ↵ To see the events between 8 p.m. and 2 a.m. on this night: *Su*) X REPORT/AFTER=4–AUG–93:20:00:0& ↵ *Su&*)/BEFORE=5–AUG–93:2:00:0 pathname ↵
/AIR	Reports air flow faults, as detected by the SCP (on MV/8000s with an SCP diskette only). This switch is relevant for error log files only. The event code is 102 (146).
/ATU	Reports events in which the CPU Address Translation Unit (ATU) accelerator was disabled or enabled by an operator command (detected by the SCP). Normally, the ATU accelerator is enabled. An ATU disable/enable cannot occur on an MV/4000. This switch is relevant for error log files only. The event codes are 122 (172) and 123 (173).
/BB	Reports successful transfers to battery backup, as they were detected by SCP. This switch is relevant for error log files only. The event code is 104 (150).
/BEFORE=*dd–mon–yy [:hh:mm:ss]*	Reports entries that occurred on or before the *dd–mon–yy* or *hh:mm:ss* specified date and time. This switch is relevant for system and error log files.
/BNC	Reports the first word of buffer–not–clear events (the SCP did not clear the host–SCP buffer as expected). This switch is relevant for error logs only. The event code is 80 (120).
/BT	Reports SCP time–outs (no response from SCP detected by the operating system within 20 seconds). This switch is relevant for error logs only. The event code is 77 (115).

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/C	Reports all CPU–related information. It includes all information gathered by the following switches: /AIR, /ATU, /BB, /BNC, /BT, /DT, /FATAL, /HANG, /HI, /HP, /HRCC, /IC, /IOC, /IPF, /IPR, /LD, /NF, NRCC, /OS, /PAR, /PH, /PW, /PWR, /RB, /RC, /REC, /RES, /RQ, /SB, /SC, /SCP, /TE, /UN, /UPSC, and /XCM switches. Most of these reports will be *NO xx EVENTS RECORDED*. This switch is relevant for error logs only.
/CHGUSER	Reports process create events that change username, displaying both the old and the new usernames. This switch is relevant for system logs only. The event code is 910 (1616).
/CONSOLES=*n*	Reports on the connect time (/CT) of a particular terminal or a group of terminals. Enter a list of terminal port names separated by + (plus) of up to 512 characters in length. An empty list will be ignored. The difference between the /CONSOLES= and /CT switches is that /CT will report on all terminals, whereas /CONSOLES= will report on a specific group of terminals (up to a maximum of 300).
/CT	Reports Connect Time (CT). For each terminal user, this report gives username, connect time, and terminal name (terminal name information is not in the default report). This switch is relevant for system logs only. The event code is 1024 (2000).
/DE	Reports device errors (DE). Reports all device errors, including unsolicited errors, in the logfile. Lists the date and time, device code, unit number, status word, the number of retries attempted, and the error type: hard or soft. Can be used independently, or with the /DX switch. This switch is relevant for error logs only. The event codes are 4 (4) and 141 (215).
/DEVICES=dd*[.u]*	Reports device errors for specific devices, where dd is the device code and *u* (optional) is the unit; for example, 27.0 for DPF0. This switch cannot report unsolicited errors (event code 141 (215)), if you specify *u*, from H.A.D.A./MV or CLARiiON disk arrays. The default error log report always includes this information. This switch is relevant for error codes only. The event codes are 4 (4) and 141 (215).

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/DT	Reports error and normal returns from DTOS, as detected by the SCP (on machines with an SCP diskette only). Do not try to run DTOS while the operating system is running. The operating system does not record this event. The event codes are 126 (176) and 132 (204).
/DX	Reports device errors extended (DX). Reports all device errors in the logfile, like the /DE switch, but provides greater detail. This switch is relevant for error logs only. Must be used in combination with the /DE switch. The event code is 4 (4).
/EBM	Reports any changes to the state of the exclusion bit map, which you can address by writing a program that uses the system call ?SYLOG (see the manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R through ?Z*). The event code is 14 (16).
/EV	Reports all human–defined events (EV). Superusers can write messages to the log file with the CLI command LOGEVENT. They can use the ?LOGEV system call instead, with the general–purpose event code 1065 (2051). REPORT will include human–defined messages only if you include the /EV switch. As with all reports, it will start on its own page. AOS/VS translates events logged by the CLI command LOGEVENT to all uppercase characters, and the /EV switch prints only the first 59 characters of each record. Under AOS/VS II, REPORT prints however many characters the user logged (up to the system limit of 496. characters), leaving case as the user typed it. REPORT prints the message, up to a system limit of 496 characters. REPORT does not change the case of the characters, but does convert each nonprinting character into a space. This switch is relevant for system logs only.
/FA	Reports a XODIAC functional level summary. This switch includes the number of FTA connections, RMA requests, FTA I/O blocks, total packets transmitted and received, and total bytes transmitted and received. Also see the /X switch. This switch is relevant for system logs only. The event code is 1067 (2053).

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/FAILED_LOGONS	Reports on all failed logons — situations where someone started to log on but did not complete the procedure. A surge in the number of failed logons may mean that someone is trying to break into your system. This switch is relevant for system log files only. The event code is 1215 (2277).
/FATAL	Reports fatal operating system errors (panics). ESD records these when you run it if logging is on. This switch is relevant for error logs only. The event code is 41 (51).
/FE	Reports XODIAC functional level errors. By username, the report includes the host, virtual circuit number, and error code for each network function that hit an error. Also see the /X switch below. This switch is relevant for system logs only. The event code is 1068 (2054).
/FILE*[=full–pathname] [+full–pathname] [...]*	Reports on all accesses, successful and unsuccessful, to all files, the file(s) named in *full–pathname,* or to the files in a directory named in *full–pathname*. You can designate a directory by adding a colon (:) to *full–pathname*. The report includes username, date, and time. You must specify a full pathname from the root directory, for example, *Su*) X REPORT/FILE=:UPD:OP ... :SYSLOG ↵ The maximum argument length is 256 characters. The /FILE switch can help you check on accesses to critical files. As with any switch, you can also specify other switches, for example, *Su*) X REPORT/FILE=:UPD:OP/TRACE=BILL :SYSLOG ↵ The event codes are 910 (1616), 912 (1624), 917 (1625), 920 through 947 (1630 through 1663), 992 and 993 (1740 and 1741), and 1214 (2276). The events represented by the codes are • File ACL change (codes 939 and 943) • File close (922) • File create (929)

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/FILE (continued)	• File delete (924 and 931) • File open (920) • File print (1214) • File rename (938 and 942) • File UDA create, delete, and write (927 and 934, 925 and 932, 926 and 933) • FSCOPY begin (992) and FSCOPY end (993) • Logical disk unit (LDU) initialize and release (937 and 928) • Permit access to protected file (used to modify shared file) (947) • Process create (910) • Process terminated by superior process (912) • Process chain to another process (917) • Shared file open (first and subsequent, 945 and 946) All of these events indicate attempts at the specified action. For example, the message *File open* means that a user tried to open a specified file. If the attempt failed, the record will include an error code which REPORT will interpret and display. If the attempt succeeded, there will be no error code. *See also* /INITIAL.
/GROUP	Reports all events where users changed their group lists. The event code is 986 (1732).
/HANG	Reports on all operating system hangs (deadlocks). ESD records this when you run it if logging is on. This switch is relevant for error logs only. The event code is 42 (52).
/HI	Reports hard interrupts from an unknown source (for example., a faulty CPU board or bad connection). The SCP records this interruption in its own log (if any), but does not send it over device code 45. This switch is relevant for error logs only. The event code is 133 (205).

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/HP	Reports host request parameter errors (in which the operating system requested SCP action, but the SCP could not comply). This switch is relevant for error logs only. The event code is 81 (121).
/HRCC	Reports MV/8000 or MV/6000 soft (single–bit) ERCC errors (replaces /ERCC switch on these machines). Relevant for error logs only. Event code is 106 (152).
/I	Tells REPORT to ignore user–defined system log entries. Use this switch to suppress *UNKNOWN CODE* messages. User programs with Superuser on can write their own codes and messages to SYSLOG, via system call ?LOGEV. (The CLI LOGEVENT command writes only to code 1065 and isn't relevant here.) If the user–defined code is not one of the standard codes, REPORT will not accept it. You should include this switch if programmers at your site have used ?LOGEV to write their own codes and messages into a log file on which you want a report.
/IC	Reports events in which the CPU instruction cache was disabled or enabled by an operator command (detected by SCP). Normally, this cache is enabled. An instruction cache disable/enable cannot occur on an MV/4000. This switch is relevant for error logs only. The event codes are 117 (165) and 118 (166).
/INITIAL*[=pathname] [+pathname][...]*	When used with switches /FILE or /TRACE, provides the username the user logged on with, even if the user has changed username. The switch ignores username changes except for those done by the EXEC process or the process(es) you specify with *=pathname*. This switch is relevant for system logs only. The event code is 910 (1616).
/IOC	Reports all IOC (input output controller) parity errors, as detected by the SCP. This has no meaning on MV/4000 logs. If the error occurs on behalf of a user, the operating system logs a hard error. If the error occurs on behalf of the operating system, the operating system panics and can't log the error. This switch is relevant for error logs only.

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/IPF	Reports infinite page faults, as detected by the SCP. The operating system will probably not stay up to record this, if it happens. This switch is relevant for error logs only. The event code is 116 (164).
/IPR	Reports infinite protection faults detected by SCP. Read the comment under /IPF. This switch is relevant for error logs only. The event code is 115 (163).
/JPRS	Job processor restart. This switch will also record the date and time of a job processor restart.
/L=*pathname*	Sends the report to the generic @LIST file, set by your CLI LISTFILE command. If you include =*pathname*, the switch sends the report to the file named *pathname*; for example., /L=@LPT. This switch is relevant for both system and error logs. If you omit it, the report goes to @OUTPUT which, for interactive use, is your terminal screen.
/LD	Reports events in which the diskette log became inoperative (MV/8000s with an SCP diskette). This switch is relevant for error logs only. The event code is 114 (162).
/LPP	Allows you to determine the page length used for pagination of the output. The default value is 63 lines per page, which effectively gives you 55 lines of data per page.
/MIRROR	Reports LDU mirroring events, including hardware synchronized, 982 (1726); hardware unsynchronized, 983 (1727); software synchronized, 984 (1730); and software unsynchronized, 985 (1731).
/MT	Reports on user mount requests. For each mount request, this report gives the username and identifies the tape(s) or disk(s). This switch is relevant for system logs only. The event code is 1025 (2001).
/NA	Reports XODIAC connection level summary. For each network user, this report shows the number of connections, connect time, packets transmitted and received, and number of bytes transmitted and received. (For reports on all XODIAC events, use the /X switch.) This switch is relevant for system logs only. The event code is 1067 (2053).

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/NE	Reports XODIAC connection level errors. For each link, this report includes each error code, associated diagnostic number, channel, virtual circuit number and transmit or receive status. A –1 for any entry means "not applicable." See also the /X switch. This switch is relevant for system logs only. The event code is 1068 (2054).
/NF	SCP interface degrade (no response received from SCP). This switch is relevant for error logs only. The event code is 78 (116).
/NRCC	Reports soft (single–bit) ERCC errors on MV/10000s and MV/4000s (replaces /ERCC switch on MV/10000s and MV/4000s). The operating system does not try to recover from, or log a record of, multibit or Sniff ERCC errors. This switch is relevant for error logs only. The event codes are 43 (53) for single–bit errors, 44 (54) for multibit errors, and 45 (55) for Sniff errors.
/OS	Reports operating system errors. This switch is unused in AOS/VS Revision 2.00 and later. This switch is relevant for error logs only. The event code is 113 (161).
/PAR	Reports parity errors on the following boards and busses (detected by SCP or other hardware): • System Cache, the event code is 108 (154); • Cache to Bank Controller, the event code is 109 (155); • IOC (I/O controller), the event code is 110 (156); • Microsequencer, the event code is 107 (153); • S–bus, the event code is 112 (160). The /PAR switch is relevant for error logs only.
/PH	Reports main processor halts, as detected by SCP. This switch is relevant for error logs only. The event code is 98 (142).
/PP	Reports pages printed. For each user, this report gives the number of pages printed per file; this information is not part of the default report. This switch is relevant for system logs only. The event code is 1027 (2003).

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/PR	Reports all privileged users. This switch gives the names of all users who are logged on and who have Superuser, Superprocess, or access devices privileges. A user need not activate the privilege to be listed. This switch is relevant for system logs only. The event code is 1026 (2002).
/PT	Reports all process terminations. For every process that was active during the log period, this switch describes the date and time of termination, full process name, and other information. The /PT switch can produce a very sizable report, since a process terminates after every text editing session, compilation, batch job, and so on. This switch is relevant for system logs only. The event code is 3 (3).
/PW	Reports CPU power failure and power restore. These are logged as one event, at the time of power restore. This switch relevant for error logs only. The event codes are 7 (7) and 40 (50).
/PWR	Reports CPU power failure and power restore. These are logged as one event, at the time of power restore. This switch is relevant for error logs only. The event codes are 100 (144) and 101 (145).
/RA	Reports a XODIAC RMA agent summary. By username, this summary gives total connect time and RMA requests. Also see the /X switch. This switch is relevant for system logs only. The event code is 1030 (2006).
/RB	Reports host–SCP buffer full faults (indicates data overflow between processors). This switch is relevant for error logs only. The event code is 76 (114).
/RC	The controller normally detects, reports, and tries to correct ERCC errors. This report includes events in which the controller detected so many ERCC errors that it disabled error reporting. (It continued to try and check/correct ERCC errors.) This event is reported via the SCP. This switch is relevant for error logs only. The event code is 131 (203).

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/REC	Reports SCP function request acknowledgements. SCP function requests may occur so often that logging them would degrade performance. The operating system does not log it. This switch is relevant for error logs only. The event code is 74 (112).
/RES	Reports SCP resets. The operating system does not log this code in SYSLOG. This switch is relevant for error logs only. The event code is 73 (111).
/RQ	Reports host–SCP request errors. The operating system does not log this code. This switch is relevant for error logs only. The event code is 75 (113).
/SA	Reports a DG/SNA accounting summary. By username, the report gives connect time, number of logical units, number of request units received and sent, and number of bytes received and sent. This switch is relevant for system logs only. The event code is 1066 (2052).
/SB	Reports S–bus time–outs (detected by SCP). The SCP cannot use the CPU S–bus. This switch is relevant for error logs only. The event code is 111 (157).
/SC	Reports events in which the CPU's system cache was disabled or enabled by operator command (detected by the SCP). This switch is relevant for error logs only. The event codes are 119 (167) and 120 (170).
/SCP	Reports status of SCP logging, as detected by the SCP. By default, SCP logging is on. Events are recorded only if someone has turned it off or on with an SCP command. This switch has no meaning for MV/4000 logs. This switch is relevant for error logs only. The event codes are 96 (140) and 97 (141).
/SECONDARY_ERROR	AOS/VS II only. Reports on secondary errors and internal inconsistency errors in the ERROR log. A secondary error is an error that occurred after a system error. An internal consistency error is a 32–bit error code; for example, an error return from a system call. The first 16 bits of an internal consistency error is ensured to be unique. REPORT will also log a record of the event with error code 11.

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/SECONDARY_ERROR continued	The format of the record is 0 – 1 The whole 32–bit error message 2 – 3 The control block address 4 – 5 The system call number 6 The PID 7 The alignment word
/SQR	System Quiesce Restart. This switch records the time and date of a restart after a normal system shutdown.
/TA	Reports a XODIAC File Transfer Agent (FTA) summary. By username, the report gives the connect time, number of FTA connections, number of FTA blocks, packets transmitted and received, and bytes transmitted and received. Also see the /X switch. This switch is relevant for system logs only. The event code is 1064 (2050).
/TE	Reports over temperature state, as detected by the SCP (on MV/8000s only). The operating system does not log this code. The event code is 103 (147).
/TRACE*[=username] [+username] [...]*	Reports on all security–related events (logons, file accesses, process creations, pages printed, and group events), that involved all users or *username(s)*, as well as multi–processor and class events. This can produce a sizable report. The report is useful when you want to monitor a user's behavior, check an application program (run by the user, like OP) for Trojan horses, or see class or multiprocessor events. Without *username*, /TRACE reports on all users; with *username*, /TRACE reports only on the named user. Maximum argument length is 256 characters. For example, *Su)* X REPORT/TRACE=JKN ... :SYSLOG ⤵ This switch is relevant for system logs only. The event codes include 910 through 947 (1616 through 1663), 966 and 967 (1706 and 1707), and 1211 through 1226 (2273 through 2312) for security–related events; 948 through 960 (1664 through 1700) and 964 (1704) for class and multiprocessor events; and 961 through 963 (1701 through 1703) and 982–985 (1726–1731) for mirroring events.

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/TRACE (continued)	The security–related events represented by the codes are • Access devices privilege turned on (call ?DEBL) or off (call ?DDIS) (code 915) • Class or logical processor event, or user locality change (951 though 960) • Terminal assigned to a program or deassigned from a program (for example, assigned to EXEC when user logs on) (918 and 919) • File ACL change (939 and 943) • File close (922) • File create (929) • File delete (924 and 931) • File open (920) • File print (1214) • File rename (938 and 942) • File UDA create, read, or write (927 and 934, 925 and 932, 926 and 933) • Group list changes (986) • Invalid log–on attempt (1215) • Job processor initialized or released, or status checked (948 though 950) • Labeled medium (tape) mount or dismount (1211 and 1212) • Logical disk initialize (937) or release (928) • Permit access to protected file (used to modify shared file) (947) • Process create (910) • Process chain to another process (917) • Process ringload (?RINGLD) of a program (916)

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/TRACE (continued)	• Process terminated by superior process (912) • Shared file open (first and subsequent, 945 and 946) • Superprocess privilege turn on/off (914) • Superuser privilege turn on/off (913) • System Manager privilege turn on/off (964) • User password change (subset of 1225) • User logon (1213) • User profile created, deleted, or renamed (1220, 1221, 1222) • User profile opened, read, or closed (for example., by EXEC when a user logs on) (codes 1223, 1224, 1226) • User profile written to (follows profile open) (1225). When anyone logs on, the time is written in the profile. Password changes are reported here and as specific events elsewhere. • Window create (966) and window delete All of these events indicate attempts at the specified action. For example, the message *File open* means that a user tried to open a specified file. If the attempt failed, the record will include an error code which REPORT will interpret and display. If the attempt succeeded, there will be no error code. If you updated your system so that it is C2–compliant, /TRACE will report on the following additional events: • Block a process (968) • Establish a connection (969) • Break a connection (970) • Change a process type (971) • Remap address space (972)

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
	• Become a system operator (974) • Set day/set time (9) • Send an IPC message (979) • Send an IPC message and wait for an answer (980) All of these events except for set day/set time require that you run SYSLOG with /DETAIL=FULL. Set day/set time will be reported with /DETAIL=MINIMAL or /DETAIL=FULL. In addition to the events listed here, there are also a number of system calls used only by the operating system or its servers. If one of these undocumented system calls is logged, the report will include the date, time, and username of the calling process, and the message *Attempted an undocumented system call*. If the user making such a call is the operating system (username=OP, or a server like DBMS, for example), a security breach is not indicated. If the report shows that a user made such a system call, you should report this fact to your security administrator. *See also* /INITIAL.
/UN	Reports unknown DIB code from SCP events. (The SCP requested the operating system to act, but the operating system could not understand or execute the request.) This switch is relevant for system logs only. The event code is 79 (117).
/UPSC	Reports UPSC (CPU power supply) faults detected by the operating system, on the following machines: MV/20000–series, MV/10000–series, MV/8000 II, MV/8000 C, and MV/4000. The operating system can record these only if it stays up through the error. This switch is relevant for error logs only.
/USERS*[=name] [+name][...]*	Reports accounting information (like CPU and I/O usage on all users or only the user(s) specified with name(s). Maximum argument length is 256 characters. For example, for a report on users F77 and SWAT: *Su*) X REPORT/USERS=F77+SWAT pathname ↵ The /USERS switch is relevant for system logs only.

(continued)

Table 11–6 REPORT Switches

Switch	What It Does
/X	Reports a XODIAC summary. This summary includes the reports from the /FA, /FE, /NA, /NE, /RA and /TA switches described above. This switch is relevant for system logs only.
/XCM	Reports occurrences of the following SCP commands, as detected by the SCP (machines with an SCP diskette only). This switch is relevant for error logs only. • BOOT (the operating system cannot record this, since it is not running.) The event code is 99 (143); • CONTINUE, the event code is 128 (200); • HALT, the event code is 127 (177); • INIT, the event code is 130 (202); • START, the event code is 129 (201); • XEQ DTOS, the event code is 125 (175).

(concluded)

SYSLOG and REPORT Examples

) SYSLOG/START ↵

... (System runs) ...

Su) XEQ REPORT/USERS=F77+SACKVILLE :SYSLOG ↵

... (Report on terminal) ...

Su) XEQ REPORT/AFTER=[!DATE]/L=@LPT :SYSLOG ↵

The preceding command reports on all entries that occurred today.

Su) XREPORT/L=COBOL_BILL/BEFORE=17-DEC-93/AFTER=12-DEC-93& ↵
Su&)/USER=COBOL:LOGS:17DEC93.LOG ↵

The preceding command reports on user COBOL between December 12, 1993 and December 17, 1993, and sends the report to file COBOL_BILL.

) SYSLOG/START/DETAIL=FULL ↵

... (System runs) ...

Su) X REPORT/TRACE=JOAN_S/L=LOGS:JOAN_S.REPORT :SYSLOG ↵

... (Report on terminal) ...

This command sequence starts the log file; then later it generates a report on all system use by user JOAN_S. The report is written to the file JOAN_S.REPORT in directory :LOGS. Superuser privilege is needed to access the current log file.

) X REPORT/FAILED_LOGONS/EV :LOGS:28SEP93.LOG ↵

This command produces a report on failed logons and events (written into the log file with the LOGEVENT command). The log file is 28SEP93.LOG in the directory named LOGS. Since no list file was specified with /L, the report is displayed on the terminal screen.

) X REPORT/X :LOGS:28SEP93.LOG ↵

This command produces a summary of XODIAC events (like those shown in Figure 11–7). Notice that a specialized report always includes the default report first, and then reports on separate pages the details for each switch. Here, the /X switch includes a number of other switches, however no events were reported for them. When detecting numbers too large to display, REPORT displays a string of asterisks (*********).

```
          23-Aug-1993     9:06:15       AOS/VS II REPORT REV 3.00.00.00

USER SUMMARY FROM FILE(S):
  :SYSLOG

USERNAME           CONNECT TIME     PAGES      CPU TIME       I/O      PAGE NUMBR
                   CONSOLE   UNIT PRINTED                   BLOCKS      SECS PROCS
*$$DAVID             59:22   0:00  262140   1:14:08.108    257740   1648418   225
*$$LIZ               38:45   0:00     105   2:50:52.470    993652   2526872   478
 $$PERFMGR            0:00   0:00       0   0:00:53.767       742     12576    15
*$$PHIL              60:28   0:00       4   1:41:51.445    186970   1827100   624
*$$RITA              14:04   0:00      94   0:20:46.126      3978    269465    40
 $JUNE               27:58   0:00  589815   0:19:25.535     12045    414247    45
 $MARG               16:01   0:00       0   0:11:08.773      4620    176030    26
 $PAT                37:32   0:00      12   1:12:56.982     42700   2143416    63
 $PRINTER             0:00   0:00     192   0:00:00.000         0         0     0
 $STEPHEN            51:15   0:00   65549   1:42:32.378     44743   3562242    59
*$STEVEN             17:27   0:00  196605   0:22:09.464     18582    581633    65
 CEO_MGR              0:00   0:00       0   0:10:00.353      8179     68111    15
 DEMO                 0:00   0:00       0   0:00:00.357         0        13     1
 DEMO.MF              0:25   0:00       4   0:02:02.171       965     22957     9
*OP                 167:19   0:00 5767089   2:31:47.277    149187   1942872   205
 OPER                 0:00  11:01      32   5:56:38.437   3376050  13702935    31
 PMGR                 0:00   0:00       0   0:38:44.045      1664    338863     1

    NO DEVICE ERRORS RECORDED.
NO DEVICE SUMMARY WILL BE PRINTED.

          23-Aug-1993     9:06:15       AOS/VS II REPORT REV 3.00.00.00

X25 SUMMARY LOG FROM FILE(S):
  :SYSLOG

USERNAME           NUMBR     CONNECT    PACKETS    PACKETS    BYTES RECD  BYTES XMTD
                   CNNCT     TIME       RECEIVED   XMTTED
$$LIZ                  8     1:05:54    262136     262136     ********    ********
OP                   103     7:12:28   2394048    2327732     ********    ********
```

Figure 11–7 XODIAC Event Summary Report from System Log

Testing System Confidence (CONTEST)

Periodically — often after generating a new system or getting new hardware — you may want to test your hardware and software.

The CONTEST confidence test package supplied with the operating system is designed to do this. It puts extreme demands on both hardware and software. If your system passes the CONTEST tests, you can be fairly confident of its ability to handle day–to–day processing. CONTEST can test

- magnetic tape unit 0;
- the master logical disk from which CONTEST runs;
- the CPU, including floating–point unit and commercial instructions;
- main memory.

You can run CONTEST on any terminal. The user process that runs CONTEST needs the following privileges in his or her profile:

Create without block
Use IPC
Use console
Unlimited sons
Change type
Change address space type
Change working set limit
Disk quota of 30000

You can create a special profile for CONTEST giving these privileges and defaulting the others, or you can edit the OP profile to contain these privileges. CONTEST also needs a PAGE and SWAP directory size of at least 75,000 blocks each. The size for the SWAP and PAGE directories will vary depending on memory size.

Running the CONTEST Package

CONTEST should not be run while time–sharing users are on the system. Run it while users are off. To run it,

- If you want to test magnetic tape unit 0, mount a scratch tape, 400 feet or more, with ring in, on the unit. Put the tape unit on line.

- Check the SWAP and PAGE directories with the SPACE command, like this

) SPACE :SWAP ↵
) SPACE :PAGE ↵

  If each is 75000 or more, proceed. Otherwise, shut down the operating system, bring it up again, override default specs, and specify 75000 for both SWAP and PAGE definitions.

- Go to a user terminal (preferably a CRT) and log on with the username under which you will run CONTEST. The profile needs the privileges described above. Your search list must include the directory :UTIL.

- Type

) CONTEST ↵

  Contest Rev n
  Memory size = x.xx Mbytes
  Do you wish to test tape unit MTx0 [N] ?

- To test your primary tape unit, type Y and press NEW LINE; otherwise press NEW LINE.

  The following tests will be run :

  ... (Names and descriptions of tests) ...

  Enter # of minutes to run Contest :

- Answer with the number of minutes you want CONTEST to run; for example, 60. An answer of 0 tells CONTEST to run indefinitely; you will then need to terminate it yourself. Sixty minutes is suggested as a general–purpose minimum time.

- The CLI prompt returns and CONTEST runs. You can let it run silently or have it summarize errors on the terminal. Messages will be recorded in disk files anyway, as described below. The summary display is useful. To get it, type

) CONTEST.ERRORS ↵

  DO NOT TOUCH THIS SYSTEM – AOS/VS II TESTING

  .
  PATH n – xxxx TEST PASSES=n ERRORS=n
  PATH n – xxxx TEST PASSES=n ERRORS=n
  .

If there are errors, CONTEST will note them in the ERRORS column and with error messages.

To see what's happening, you can run PED on this, or another, terminal by typing

) PED/ALL/CYCLE=5 ↵

- CONTEST should terminate on its home terminal when your specified period has passed. Then it will display

  Contest Run Finished
)

  If the summary shows no errors, and CONTEST terminated normally, then CONTEST detected no errors. Type CONTEST.CLEAN and press NEW LINE to delete temporary files. (Some of these are quite large, so be sure you delete them.) Type X REPORT :ERRORLOG and press NEW LINE to see if any errors occurred during the CONTEST run. If the report shows no errors, your system passed the tests; you're done.

  If you see error messages, or an *ABNORMAL PATH TERMINATION* message, then CONTEST detected an error. Proceed to the next section.

- If you want to terminate CONTEST before it's done, press CTRL–C CTRL–A to interrupt the CONTEST.ERRORS macro; wait for the CLI prompt and type BYE and press NEW LINE. CONTEST will ask for verification, and you will type Y and press NEW LINE to terminate it. Several moments may pass before CONTEST actually terminates and the CLI prompt returns. If too much time passes, you can terminate CONTEST's father CLI from the system console.

CONTEST Error Interpretation

Generally, system hangs (deadlocks) indicate software problems. System panics (FATAL AOS/VS ERROR) or CONTEST test failures mean hardware problems. If either a hang or panic occurred, the system failed the test. The display and/or messages on the system console may tell why it failed. On a deadlock, get back to the SCP, do a memory dump, and run Emergency Shut Down (ESD). On a panic, note the panic values, do a memory dump, and run ESD. Next, check the SCP. Then restart the operating system and check the error log by typing X REPORT :ERROR_LOG and press NEW LINE. The SCP log and error log may pinpoint the problem.

If the system doesn't hang or panic, you may want to check out the CONTEST error files or other error files as follows.

File CONTEST_ERRORS.TS describes CONTEST initialization errors that might occur if CONTEST lacked needed privileges like *Change type*. If this file doesn't exist, there were no such errors.

File PATH_ERRORS.TS shows a duplicate of the last summary display. It will show no errors if CONTEST found none. If there are errors, you might print (QPRINT) this file for the record.

File MONITOR.LOG.TS shows the sequences, priorities, paths and PIDs that CONTEST ran. It shows no errors but can help trace problems. Related files, which have meaning only if an error occurred, have the form

ERROR_nn_hh.mm.ss.TS

where *nn* is the path number, and *hh.mm.ss* is the time (in hours, minutes, and seconds) of the error. (Note that the leading zero of the time display is suppressed; an error occurring at 5 hours 14 minutes and 3 seconds is displayed as 5:14:03.) Each of these files represents one error on the terminal display. These files might be useful if you ran CONTEST overnight, errors occurred, and you wanted to know when the error occurred and other specifics.

File CONTEST.CLEAN.CLI is a macro that cleans up by deleting all CONTEST–created files except for CONTEST_ERRORS.TS and the ERROR+ files. Run it after checking the CONTEST and PATH files for errors. After the contents of the ERROR+ files have been noted, delete them so that the working directory is not cluttered with old information.

CONTEST is really very easy to run, and its error messages (described in the manual *AOS/VS and AOS/VS II Error and Status Messages*) are quite specific about problems.

After any CONTEST run, you should check for errors that CONTEST didn't detect. Soft and hard errors are displayed on the system console. They are also recorded in the error log. So you should always check the error log (type X REPORT :ERROR_LOG and press NEW LINE) for device errors produced during a CONTEST session.

Specific Tests and Script Files

CONTEST allows you to run specific tests and/or create your own test scripts.

To run only selected tests, for example, to test only your tape hardware or floating point, use the /Q switch:

```
) CONTEST/Q ↵
```

CONTEST then asks you which tests you want to run. To choose a test, press NEW LINE (or type Y and press NEW LINE). To skip the test, type N and press NEW LINE. After you select or reject all tests, CONTEST starts the one(s) chosen. For an error display, you can type CONTEST.ERRORS and NEW LINE as shown above.

To create your own script file without running tests, use the /N switch. CONTEST then asks about the magnetic tape. After receiving an answer, it builds a script file including all tests (except for the magnetic tape test, if you said N to magnetic tape); then CONTEST terminates. The script filename is CONTEST.SCR. If a file named CONTEST.SCR already exists in the working directory, CONTEST deletes and recreates it.

You can create a custom script file by combining the /Q and /N switches. If you combine them, CONTEST asks questions about each test. Then it builds a script file that specifies only the tests you chose. The script filename is CONTEST.SCR. If you create different script files, you can rename each after it is created so that CONTEST doesn't delete it when you create another script file. To use a script file, give the command sequence CONTEST script–filename; for example, type

) CONTEST CONTEST.SCR ↵

CONTEST Example

This example assumes the process running CONTEST has the needed privileges. Log on an EXEC–owned terminal. Put a write–enabled tape on unit 0; put the unit on line.

) CONTEST ↵

Contest Rev n
Memory Size = 2.00 Mbytes
Do you wish to test tape MTx0 [N] ? Y ↵

The following tests will be run:
PATH 70 – MV/ECLIPSE MEMORY TEST
PATH 71 – SPACE MEMORY EXERCISER
...
Enter number of minutes to run Contest : 60 ↵

AOS/VS II CLI32 Release 03.00.00.00 ....
) CONTEST.ERRORS ↵

DO NOT TOUCH THIS SYSTEM – AOS/VS II TESTING
AOS/VS II REV n
CPUID = n
PATH ERROR INFORMATION
ELAPSED TEST TIME n HOURS n MINUTES

PATH 70xx – MV/ECLIPSE MEMORY TEST	*PASSES=n*	*ERRORS=n*
PATH 71xx – SPAGE MEMORY EXERCISER	*PASSES=n*	*ERRORS=n*
PATH 72xx – SYSTEM DISK TEST (BLK I/O)	*PASSES=n*	*ERRORS=n*
PATH 73xx – TAPE EXERCISER	*PASSES=n*	*ERRORS=n*
PATH 74xx – FLOATING POINT TEST	*PASSES=n*	*ERRORS=n*
PATH 76xx – COMMERCIAL TEST	*PASSES=n*	*ERRORS=n*

where *xx* is the number of each test type currently running at any moment.

... (Display above cycles every 45 seconds) ...
... (60 minutes pass) ...

Contest finishes.

) TYPE CONTEST.ERRORS_TS ↵
WARNING: FILE DOES NOT EXIST...

) TYPE PATH_ERRORS.TS ↵

... (Duplicate of last display) ...

If there were no errors, then type CONTEST.CLEAN and press NEW LINE.

DELETED ...
DELETED ...

) X REPORT :ERROR_LOG ↵

... (Check for new device errors) ...

Monitoring ECLIPSE–bus Disk I/O (DISCO)

The DISCO disk monitor program displays statistics on disk I/O for all disks except for those connected to an MRC. (To gather disk statistics for devices connected to an MRC, you need to use the Monitor program, which is part of the AOS/VS and AOS/VS II Performance Package.) These statistics include requests made to the unit, percentage of busy time, queue length, average seek distance, and blocks read and written, among others. For systems with multiple disks, this information can help you decide when to redistribute the file system for better load balancing or to purchase an additional controller. For any system, you can use the average seek figure to detect when file fragmentation has occurred.

Read the following sections to learn how to execute DISCO, look at its screens, issue commands while DISCO runs, and use DISCO productively.

How to Execute DISCO

To run DISCO, put :UTIL in your search list and type

) XEQ DISCO ↵

Exit from DISCO by pressing CANCEL/EXIT (F11).

You can append any of the following switches to DISCO. You can abbreviate all switches to the unique form.

/BATCH — Runs DISCO in batch mode, freeing up your terminal. DISCO ignores runtime commands when you use this switch. Use /BATCH with /SNAP. For example,

) PROC/DEF DISCO/BATCH/CYCLE=1/SNAP=10/LIST ↵

/CYCLE=*n* — Defines the DISCO update cycle in seconds. The default cycle is 10 seconds. For *n*, you can specify any value from 1 to 65535. This switch is useful for longer than normal cycles, which can give you a better picture of DISCO rates. For example,

) X DISCO/CYCLE=30 ↵

/LISTFILE*[=path]* — Sends DISCO output to the file named in *path*. If you omit *path*, this switch sends output to the default list file DISCO.LIST. The file will be deleted and then recreated if it already exists. If you use this switch, DISCO displays the word "Listing" in the upper left corner of the screen. You can also enable/disable the listing during a session by typing T and pressing NEW LINE.

/MIRROR — Displays the actual statistics for an LDU that is a mirror of another LDU.

/RATE — DISCO can display two screens: the main screen with numbers accumulated since system startup, and a rate–oriented screen with data from the last cycle. To start DISCO with the rate screen (instead of the main screen), use this switch.

/RESET	By default, DISCO uses counts that have accumulated since each disk was initialized. This switch tells DISCO to start with 0 for each count. It's useful for tracking information from a specific time.
/SNAP=*n*	Collects DISCO data for *n* times. See /BATCH for how to use this switch.

DISCO Commands

While DISCO is running on your screen, you can type any of the following commands to it:

Command	What It Does
< or >	Increases (>) or decreases (<) cycle time by 1 second each time you press the key. The default cycle time (if you don't start DISCO via X DISCO/CYCLE=n) is 10 seconds.
^ or V	Scrolls up (^ Shift–6) or down (V) a line. This command is useful when more than 10 disks are initialized and you want to check one whose data doesn't appear.
?	Gives help; displays DISCO commands.
S	Displays the other DISCO screen.
T	Enables or disables listing.
Z	Zeroes the counters; DISCO starts accumulating numeric data from scratch. This command is useful when you want to accumulate data from a specific time.

If You Have a CLARiiON or H.A.D.A./MV Disk–Array

If you have a CLARiiON or H.A.D.A./MV disk–array, DISCO cannot report unit–level usage with RAID–0, RAID–3, or RAID–5 groups properly. Although DISCO can report response times correctly, Avg/Max Queue length, % busy, and Avg Serv time will be misleading. You should use the GridMgr utility at the storage–system console as well as the AOS/VS II Performance Monitor to gather statistics for these disk–array storage systems.

The DISCO Screens

There are two DISCO screens, which are quite similar. Figure 11–8 shows the first DISCO screen that displays total figures since system startup. Figure 11–9 shows the second DISCO screen that displays figures from the last cycle only.

```
May 04,1993          AOS/VS II  DISC  MONITOR  PROGRAM                    11:28:18
                                                          Function key #11 to Exit

Actual time      0.0 seconds                                Cycle time     10 seconds

     U
  D  n                                                                      Avg   Avg
  e  i    # of    % of   % of  % of   Avg/Max  Avg    Blocks    Blocks    % of Serv  Resp
  v  t    Reqs    Total  Busy  Intf    Queue   Seek    Read     Written   Util Time  Time
-------------------------------------------------------------------------------------
 24  0  1452821   43.3   72.2   0.0   1.60  10 393.7  2688634   2985444  36.9 .041  .067
 24  1   121619    3.6   17.1   0.0   1.10   3 135.9  1311696    114801   2.7 .037  .041
 24  2    13584    0.4   26.8   0.0   1.14   2 178.7   365498      3017   0.3 .046  .053
 24  3    65666    1.9   27.9   0.0   1.19   3 118.6  1107189     2234   1.6 .042  .050
 64  2   178353    5.3    9.9   0.0   1.05   3 221.3  1642821    314967   3.8 .035  .036
                  ============= second IOC ==========
126  0      292    0.0    0.0   0.0   1.00   0 287.1     1504       142   0.0 .041  .041
126  1   482523   14.4    9.8   0.0   1.06  10 164.1  2851807    432120   9.3 .031  .033
126  2   230000    6.8   14.2   0.0   1.07   3 103.9  2505551    148001   4.4 .031  .034
166  1   803527   23.9   29.1   0.0   1.34   7  59.4  6343722    728629  14.8 .030  .040
166  2      175    0.0    0.0   0.0   1.00   0 358.1      485        95   0.0 .045  .045
```

Figure 11–8 First DISCO Screen

Actual time 60.4 seconds Cycle time 60 seconds

Dev	Unit	# of Reqs	% of Total	Reqs per sec	% of Busy	% of Intf	Avg/Max Queue	Avg Seek	Blocks/sec Read	Blocks/sec Write	% of Util	Avg Serv Time
				Per Cycle								
24	0	743	13.1	12.3	53.0	0.0	1.59/ 8	190.7	80.9	14.5	52.4	.043
26	0	90	1.5	1.4	1.1	0.0	1.00/ 6	426.3	7.2	3.9	4.9	.033
26	1	3	0.0	0.0	0.0	0.0	1.00/ 2	309.6	0.7	0.0	1.6	.333
27	0	159	2.8	2.6	4.4	0.0	1.02/ 6	41.4	28.2	0.4	4.9	.018
27	1	5	0.0	0.0	0.0	0.0	1.00/ 3	290.8	1.3	0.0	0.0	.000
27	2	0	0.0	0.0	0.0	0.0	0.00/ 3	0.0	0.0	0.0	0.0	.000
64	0	558	9.8	9.2	10.5	0.0	1.06/ 4	181.6	12.9	26.1	60.0	.064
64	1	747	13.2	12.3	11.5	0.0	1.06/ 9	106.1	10.8	46.1	71.6	.057
64	2	832	14.7	13.7	93.9	0.0	5.60/25	51.0	1.1	76.4	75.0	.054
64	3	303	5.3	5.0	5.2	0.0	1.03/ 4	90.6	48.1	1.5	30.0	.059
66	0	0	0.0	0.0	0.0	0.0	0.00/ 2	0.0	0.0	0.0	0.0	.000
67	1	28	0.4	0.4	0.0	0.0	1.00/ 2	123.7	6.8	0.0	0.0	.000
67	2	0	0.0	0.0	0.0	0.0	0.00/ 3	0.0	0.0	0.0	0.0	.000
		========== second IOC ==========										
124	0	10	0.1	0.1	0.0	0.0	1.00/ 3	238.6	1.8	0.0	0.0	.000
124	1	10	0.1	0.1	0.0	0.0	1.00/ 2	24.6	2.3	0.0	3.3	.199
126	0	385	6.8	6.3	4.6	0.0	1.02/ 7	28.5	30.5	7.6	16.6	.025
126	2	0	0.0	0.0	0.0	0.0	0.00/ 2	0.0	0.0	0.0	0.0	.000
127	0	204	3.6	3.3	5.3	0.0	1.02/10	227.3	37.0	1.3	8.3	.024
127	1	95	1.6	1.5	0.0	0.0	1.00/ 6	203.8	7.8	2.0	5.0	.031
127	2	2	0.0	0.0	0.0	0.0	1.00/ 3	8.0	0.2	0.0	0.0	.000
127	3	83	1.4	1.3	3.6	0.0	1.01/ 5	56.2	15.2	0.1	8.3	.060
164	0	587	10.3	9.7	23.8	0.0	1.15/ 7	296.0	60.0	21.4	45.0	.045
164	1	33	0.5	0.5	0.0	0.0	1.00/18	242.4	7.4	0.0	1.6	.030
164	3	87	1.5	1.4	13.7	0.0	1.06/ 4	206.7	12.8	0.4	8.3	.057
166	1	15	0.2	0.2	0.0	0.0	1.00/ 3	98.0	3.1	0.0	1.6	.066
167	0	75	1.3	1.2	7.9	0.0	1.03/ 6	50.8	13.2	0.2	5.0	.039
167	1	40	0.7	0.6	0.0	0.0	1.00/ 3	96.9	10.5	0.0	0.0	.000
167	2	559	9.8	9.2	22.0	0.0	1.13/ 7	48.9	126.9	1.7	28.3	.030

Figure 11-9 Second DISCO Screen

DISCO can display on the screen information about 10 disk units at once. It describes all initialized disks. When a disk is released, DISCO stops describing it.

On the second screen, DISCO includes the columns "% of Util" and "Avg Serv Time" only if the cycle time is 60 seconds or more. The reason for this is that a shorter cycle time would not produce a sufficient number of samples.

If the system has disks on a second I/O Controller (IOC), DISCO displays the secondary IOC disk information after the primary IOC disk information.

What DISCO Column Heads Mean

The DISCO columns have the following meanings (listed alphabetically, with the # and % symbols first):

Column Head	Means
# of Reqs	Shows I/O requests made to the unit. On the main DISCO screen, this column shows total requests since the disk was initialized (or since you pressed the Z key). On DISCO's rate screen, this column shows requests in the last DISCO cycle. (This distinction also applies to numbers under other column heads that appear on both screens.) On a system with multiple disks, during memory contention, the disk(s) that holds directories SWAP and PAGE often shows more accesses and a higher percentage of I/O than others.
% of Busy	The percentage of I/O requests made to this unit while one or more requests were already queued to it. It appears on both DISCO screens (on the second screen only if the cycle is 60 or more seconds). Comparing "% of Busy" to "% of Util" (first screen) can be useful. If "% of Busy" is relatively high and "% of Util" is relatively low, this means the disk load tends to spike; it's not constant. This may mean that an application program is too frequently accessing the disk — a situation that you might be able to improve after discovering why the program is doing this. If both "% of Busy" and "% of Util" are relatively high, you might want to consider moving part of the load on this unit to another unit.
% of Intf	The percentage of requests to this unit that had to and wait at the controller level because the disk's % of Intf controller was processing another request. It appears on both DISCO screens. The interference number should be very low — below 5%. If it exceeds 5%, you should consider putting the unit on its own controller.
% of Total	The percentage of all disk I/O that was performed by this unit. It appears on both screens.
% of Util	The percentage of time the disk was found busy. The system checks a disk status flag every second; this number indicates the percentage of times the flag indicated busy. (This column relates to "% of Busy" above.) This number appears on both DISCO screens, but only on the second screen if the cycle time is 60 seconds or more.

Column Head	Means
Avg Queue	The average queue length for this unit. (The average number of requests at any moment.) It appears on both screens.
Avg Resp Time	The service time (described next) plus time spent waiting in the disk queue.
Avg Seek	The average number of cylinders crossed in a seek to fulfill a read or write command. It appears on both DISCO screens. Seek distance indicates two important things: the amount of free disk space available and the amount of file fragmentation. Large figures show a relatively full disk and/or a high level of fragmentation.
Avg Serv Time	The average time in seconds for each request from reaching the head of the queue to completion. It appears on both DISCO screens, but only on the second screen if the cycle time is 60 seconds or more.
Blocks Read	Shows total disk blocks read from the unit.
Blocks/sec	The Read and Write columns show the average number of disk blocks read from and written to the unit, during the last DISCO cycle. It appears on the DISCO rate screen.
Blocks Written	The total disk blocks written to the unit.
Dev	The device code of the disk controller (octal). It appears on both screens.
Max Queue	The largest number of I/O requests ever queued to this unit at once: the longest queue it has had. It appears on both screens.
Read	See "Blocks/sec."
Reqs per sec	The average requests per second during the last DISCO cycle.
Unit	The unit number of the disk. It appears on both screens.
Written	See "Blocks/sec."

DISCO accumulates data as each request is made. Its data pool includes information gathered during all requests since the disk was initialized. Since data shows status during requests only, DISCO percentages may not give a purely objective picture of disk activity. (These percentages apply to average queue figures).

If you have logical disk mirroring, note that DISCO cannot report average seek distances. DISCO will report the combined statistics for each element of the mirror. In order to keep the percentages accurate, DISCO counts only one copy in the totals it uses for computational purposes.

Also, DISCO reports statistics on the mirrored — not the mirroring — logical disk. For the mirroring logical disk, DISCO displays the message *Mirror of device n unit m,* where *n* is a 2 or 3 digit device code and m is 0 through 3.

Using DISCO Productively

If your system has multiple disks, DISCO information can help you balance disk load and distribute files for best performance.

In any system, DISCO's "Avg Seek" figures can help you check the impact of disk fragmentation. If the "Avg Seek" figures grow over time, you might consider rebuilding your logical disk(s) to reduce fragmentation.

DISCO's "% of Busy" and "% of Intf" can also tell you, respectively, about the constancy of load on a disk, and whether you should consider an additional controller.

To get a better understanding of DISCO, start it up. Then make sure that all needed system software is running and all pertinent applications software is ready to run. Type Z to zero DISCO numbers. Then start and run your applications software as usual.

With multiple disks, DISCO figures can help you equalize processing load, perhaps by moving often–used files. In any system, if you create a base of reference, the average seek figures can help tell you whether file fragmentation is hurting performance.

One way to create a base of reference is to create a standard disk–exercising procedure. For example, try a procedure that involves creating three directories, moving all Help files — :HELP: — to them, and having three user processes issue F/AS/S commands in these directories. While all this is going on, run DISCO and check the "Avg Seek" figures. Afterward, delete the directories. Ideally, you'd try the disk–exercising procedure when your disks were new (no fragmentation). Later, whenever you suspect fragmentation, you would run the procedure again. If the average seek numbers are larger with each run, file fragmentation is probably slowing down your system. There is more information on fragmentation in Chapter 13, "Fine–Tuning System Performance."

Displaying AOS/VS II Disk and LDU Information (LDUINFO)

The AOS/VS II LDUINFO utility can provide information on all physical disk units and on all LDUs in your system. LDUINFO is an interactive menu–driven program with on–line, context–sensitive help. By default, LDUINFO reports on all LDUs — initialized and not initialized — in your system.

To acquire physical information on an LDU, you need Execute access to :PER and Read access to the disk unit(s); to acquire logical information on an LDU, you need Execute access to its parent directory(ies) and Read and Execute access to the LDU filename. In any case, you can overcome all restrictions by turning Superuser on.

Depending on how you run it, LDUINFO reports physical or logical information, or both. Generally, use physical information to tell about the LDUs on a given unit; use logical information to tell whether an LDU has been initialized, how large it is, and whether the LDU is currently mirrored.

Physical Information Reported

Physical information includes the following LDU information for each physical disk you specify. LDUINFO reports this information whether or not any LDU on the disk is initialized. For each LDU it finds on the disk, LDUINFO reports the following information.

- LDU name;
- LDU unique ID;
- Whether the LDU is incomplete (LDUINFO could not find all pieces, perhaps because a unit is off line or you do not have access to the LDU), inconsistent (two or more pieces of an LDU exist with the same name and same LDU unique IDs, perhaps because of a mistake running Disk Jockey), or most recent;
- Names of all physical disks that hold pieces of the LDU (if an LDU spans more than one physical disk);
- Whether the physical disk holds more than one LDU;
- Piece number, in the form *Piece n of n*; n is 1 if the LDU has one piece;
- System area ID of the LDU;
- Size of the disk piece;
- Size of the disk bad block table (BBT);
- Starting disk address of the LDU (if there is more than one LDU on the disk, the starting address of the first LDU);
- Modify time. If an LDU is mirrored, the modify time is the time the last synchronization completed; the system uses this time to determine the preferred LDU image. The image with the later modify time is usually the preferred image.
- If an ECL disk, the channel number, device code, and unit type; and,
- If an MRC disk, the node number and the IOC device code.

Logical Information Reported

Logical information includes the following LDU information for each LDU you specify.

- Whether or not the LDU is initialized. If the LDU is not initialized, LDUINFO displays the error message *File does not exist*; LDUINFO cannot display other logical information. You can also issue the command F/AS/TYPE=LDU from the root directory, which is where LDUs are normally initialized, to check for initialization.
- Whether or not the LDU was initialized for data caching (initialized with the INITIALIZE/CACHE switch or, for the master LDU, whether caching was specified at system startup). If the LDU was initialized for data caching, LDUINFO displays the word *cached* after the LDU filename. For example,

  UDD1 [cached]
- If the LDU is mirrored. For example,

  UDD1 [mirrored]
- If the LDU is mirrored and an image was removed. For example,

  UDD1 [an image was removed]
- If the LDU is mirrored and the system is synchronizing its images. If the LDU images are being synchronized, LDUINFO displays the words *sync in progress* after the LDU filename. For example,

  UDD1 [sync in progress]

  If the LDU is mirrored and the images are fully synchronized, *sync in progress* does not appear, but the image unique IDs are displayed under the header *active images* near the bottom of this screen.
- Size of the LDU (in 512–byte disk blocks)
- Cache statistics, if the LDU was initialized for data caching. The cache statistics display shows the number of reads and writes and cache hits, in the following form

  Cache statistics:

Reads:	*n*	*Read hits: n*
Writes:	*n*	*Write hits: n*

  If the LDU was initialized for data caching, this information tells you how efficiently the cache is working.
- Time the LDU was initialized.
- Default LDU parameters, including primary and secondary data element sizes, number of primary elements, index element size, and maximum number of index levels.
- Active images. This shows the unique IDs of all synchronized LDU images; it tells you if the LDU is currently mirrored and synchronized with another image; whether hardware mirroring is in effect; and whether each active image is primary. If LDUINFO displays more than one item under this header, you know that the LDU is currently mirrored.

To discover which screen gives the information you want, consult the alphabetical list of items in Table 11–7, next.

Table 11–7 How to Obtain Information from LDUINFO

For Information on This Item	See This Screen Entry
Any Physical Disk item	None. Specify a full report, and @LPT or a disk file.
Bad block table	Physical information on disk, "BBT" entry.
Cache	Logical information on LDU, "LDU" entry. This shows *cached* if the LDU is cached. Logical information on LDU, "Cache statistics" entry. This shows the number of reads and writes, with cache hits, if the LDU is cached.
Disk	Physical information on disk.
Initialization status	Logical information on LDU. If the system does not report *File does not exist*, the LDU is initialized. (Or the LDU is initialized, but not in the root.)
LDU	
filename	Physical information on disk, "LDU" entry.
general	Logical information on LDU, all entries.
pieces	Physical information, "Piece" entry.
unique ID	Physical information, "LDU unique ID" entry.
LDUs on physical disk	Physical information on disk.
Mirror	
synchronization	Logical information on LDU, "LDU" entry. This tells if images are being synchronized. Logical information on LDU, "active images" entry. This tells the unique IDs of all synchronized images.
most recent image	Physical information on disk that holds LDU, "Modify time" entry. This tells the time the LDU image was last synchronized. The image synchronized later is usually the preferred image. If the images of a mirror are not currently synchronized or being synchronized, and you want to know which units hold the images, use the physical information screen for as many units as needed and look for the same LDU filename. All mirror images have the same LDU filename.
Modify time of image	Physical information on disk, "Modify time" entry.
Pieces of an LDU	Physical information on disk, "Piece" entry.
Size	Logical or physical information, "Size" entry.

Running LDUINFO

The LDUINFO program makes extensive use of function keys; you should run it on a video display terminal, not on a hardcopy terminal. Generally, LDUINFO function keys work the same way as other AOS/VS II utility programs such as VSGEN; you can use the AOS/VS and AOS/VS II Menu–Based Utilities template with LDUINFO. Use Shift–F1 to get help as needed. You can abort LDUINFO by pressing CTRL–C CTRL–A or CTRL–C CTRL–B.

The LDUINFO program file, LDUINFO.PR, is shipped in directory :UTIL. Run LDUINFO using the format

XEQ LDUINFO/*L[=pathname] [disk–unit–name* or *ldu–filename] [...]*

If you omit arguments, LDUINFO will run interactively; it will prompt for the type of report you want, and then it will gather information on disk units and LDUs you specify. You can specify a file other than your terminal if you want; therefore if you want to print a report, omit arguments.

If you include one or more arguments, LDUINFO reports on each item you specify. For each disk unit name you specify, LDUINFO displays physical disk information; for each LDU filename you specify, it displays logical LDU information. The switch */L[=pathname]* writes information to pathname or to the generic list file @LIST.

Sample LDUINFO Dialog

The following example shows LDUINFO run without an argument; selections are made via interactive dialog. This system has four LDUs on four disks. The LDUs are named ROOT, UDD1, CEO, and CEO_FILES.

Su) X LDUINFO↵

... (program displays introductory information, including warnings) ...

Specific or full physical report (S or F)? [S] ↵	(Press NEW LINE to select specific report.)
LDUINFO Rev n *LDU Display* *Page 1 of 1*	
F1 get LDU info *Shift–F1 help* *F3 page back* *F4 page forward* *F11 Cancel / Exit* *Cursor arrows change selection*	(Program lists valid function keys.)
LDUs found in system:	
: :BACKUPS :CEO :UDD1 –> F1	(Use rightarrow key to select UDD1; then press F1 to get information on :UDD1.)
Do you want Logical or Physical information on :UDD1 (L or P) [L]?↵	(Select logical by pressing NEW LINE.)

Logical Information Display

If you choose Logical information, LDUINFO displays the following screen:

LDUINFO Rev n *Logical LDU Information Display*

Shift–F1 help F11 Cancel / Exit (Program lists valid function keys.)

UDD1 [cached] (LDU was initialized for data caching.)

Size: 1144392 blocks (Size is 1,144,392 disk blocks.)
Cache statistics: (Cache statistics show a hit rate of greater than 50 percent on reads.)
Reads: 37804 *Read hits: 20349*
Writes: 25807 *Write hits: 11304*

Time initialized: Wed Aug 18 08:58:02 1993 (Date LDU was initialized)

Default LDU parameters:

Primary element size:	*4*	*Secondary element size:*	*4*
Number of primary elements:	*1*	*Index element size:*	*1*
Maximum index levels:	*3*		

Active image(s):
UDD1.IMAGE1 – DPJ11 (Just one image; LDU is not mirrored.)

Physical Information Display

When you press P and NEW LINE, the LDUINFO program — if a mirrored LDU's images have the same name — LDUINFO first asks you to choose based on LDU unique ID. LDUINFO does this through its LDU Unique ID Display. In this example, there are two LDUs with the same name, TEMP:

LDUINFO Rev n *LDU Unique ID Display*

F1 select LDU unique ID Shift–F1 help (Program lists valid function keys.)
F11 Cancel / Exit Cursor arrows change selection

LDU: *TEMP* (One LDU filename with *two* LDU unique IDs; choose the one you want by pressing NEW LINE or downarrow and NEW LINE.)
LDU unique IDs:
TEMP
REV200_TEMP

If you choose Physical information, LDUINFO displays its Physical LDU Information Display. The following example is for the first system.

LDUINFO Rev n *Physical LDU Information Display*

Shift–F1 help F11 Cancel/Exit		(Program lists valid function keys.)
LDU:	*UDD1*	(LDU filename.)
LDU unique ID:	*UDD1.IMAGE1*	(LDU unique ID.)
Disk unit(s):		
@DPJ11		(Disk unit(s) that holds pieces of LDU. If more than one, use cursor keys to select the piece you want.)
Piece 1 of 1	*System area ID: 500*	(Piece 1 of a 1–piece LDU.)
Size: 1144392 blocks	*Channel number: 0*	(LDUINFO reports on size
BBT size: 256 blocks	*Device code: 64*	of piece, BBT, and other
Starting LDU address: 0	*Unit number: 1*	physical information.)
Unit type: DPJ	*Modify time: Wed Aug 18 08:58:02 1993*	

LDUINFO Summary

Use LDUINFO whenever you need information on the state of a an LDU or disk unit: LDU mirror status, how many disk units an LDU occupies, whether it was initialized for data caching, the LDU unique ID, or other information. The Disk Jockey program can give you some of this information, but LDUINFO is easier to use because it works with LDUs that are initialized.

Changing Program Preamble Parameters (SPRED)

The Selective Preamble Editor (SPRED) program allows you to change the behavior of a program when it runs. Using SPRED, you can

- change paging or swapping parameters;
- change PID–size type; and
- change program locality.

SPRED allows you to edit a program's preamble. The system reads the preamble when it initializes the process in which the program runs. SPRED by itself doesn't change the process's behavior; other system parameters allow or enable these actions.

Changing Paging or Swapping Parameters

The paging and swapping operations you can specify with SPRED are

- Loading one or more program pages into memory when the program is executed. The VSGEN questions that enable loading program pages into memory are *Max program load pages–noncontention* and *Max program load pages–contention.*
- Adding two or more pages to the process working set when the process takes a page fault. The VSGEN question that adds two or more pages to the working set is *Fault time prepaging maximum.*
- Maintaining a nonstandard size swap file for the process. The VSGEN question to allow a process to maintain a nonstandard size swap file is *Do you want to use variable swapfiles*, followed by questions that set a maximum and a default size.

No value set with SPRED can override the value given in the corresponding VSGEN question. For example, if the value for the maximum program load pages is 35 and the default VSGEN number is 30, the system will ignore the SPRED setting for it.

The paging operations you specify with SPRED can speed up a program with many unshared pages if the program takes many unavoidable page faults. An example is a program with large, static data tables built in. By having the system load many pages at once or swap without needing to reduce a working set, you reduce overhead, since some faults do not occur. Also, the system can optimize the reading of pages from disk, thus improving process response.

Like all performance tools, SPRED requires care. If a program is faulting because of memory contention, SPRED operations won't help and may hurt. (In memory contention, reducing the contention is more effective than any SPRED–related operation.) In addition, SPRED cannot reduce shared–page overhead.

Changing PID–size Type

Before AOS/VS Revision 7.00, programs ran as processes with PIDs up to 255. Later revisions of AOS/VS and AOS/VS II permit processes to run with PIDs greater than 255. You may not need this feature, called big PIDs, on your system, but on large systems with many user processes, you may want to use big PIDs.

To allow big PIDs, a system must have been generated with a nondefault answer to the VSGEN parameter question *Maximum number of processes*. You may also have to change a program's PID–size type.

SPRED allows you to view or change a programs PID–size type. Briefly, the three PID–size types are

smallPID — A program that can run only with a PID below 255.

hybrid — A program that can run only with a PID below 255 but can create sons below and above 255. The CLI, EXEC, and most other operating system programs are hybrid programs.

anyPID — A program that can run above 255 and create sons above and below 255.

Before changing a programs PID–size type, read "Running More than 255 Processes on Your System" in Chapter 13, Fine–Tuning System Performance. That section explains some reasons for and consequences of changing PID–size type.

Changing Program Locality

When the operating system schedules a process to run, it looks at the process priority and other factors. If you have purchased the optional Class Assignment and Scheduling Package (CLASP) software (or use class system calls and write your own program), you can alter the scheduling of processes.

Briefly, CLASP lets you define classes for processes. A class is a set of processes for which you want special scheduling treatment (usually, some percentage of processing time). Each class has one or more user localities (set by PREDITOR) and one or more program localities (set by SPRED). A process will run in a class if its user and program localities match those defined (with CLASP) for the class.

SPRED Format and Switches

The SPRED command line has the form

XEQ SPRED */ARGFILE=argfile]* */DISPLAY* */LOGFILE=logfile* */S=symtable*
/D */LOG=logfile programfile*

Everything except XEQ SPRED is optional; SPRED will ask interactively for the information it needs. If you use the /ARGFILE, /DISPLAY, or /D switch, you must specify the program filename. The switches have the following meanings:

/ARGFILE=argfile

Tells SPRED to take settings from the filename (pathname) *argfile* and write them into *programfile*. *argfile* must be an argument file created during a previous SPRED run. Argument files allow you to apply SPRED edits noninteractively (even in batch, via QBATCH if desired). For example,

```
) XEQ SPRED MYPROG.PR ↵
```

(After you change settings, choose option 9 to create an argument file.)

```
)
```

(Time passes. Someone relinks MYPROG.PR, overwriting the SPRED–edited version.)

```
) QBATCH XEQ SPRED/ARGFILE=MYPROG_1.ARG& ↵
&) MYPROG.PR ↵
```

This sequence shows how to use an argument file to implement SPRED edits easily.

For a display of the program's preamble, also include the /D switch.

/DISPLAY or */D*

Tells SPRED to display the program's preamble and terminate.

/LOGFILE=logfile or */LOG=logfile*

Tells SPRED to create (or append to) the file named in *logfile*. SPRED records all dialog just as you see it on the screen.

/S=symtable

Names the program symbol table (to resolve symbolic references). This switch is needed only if the symbol table has been renamed or moved.

programfile

Identifies the program (.PR) file whose preamble you want to edit.

During a SPRED session, SPRED displays settings in brackets, followed by (from session) or (from file) if someone changed the original setting in a previous SPRED session. If the original setting is unchanged, it displays the value as none. For example, the first time you choose to edit the swap file setting, SPRED displays

Swap file size (in decimal): [none] (from session)
New value:

If you type a new value, say 400, then during this session, SPRED will display the swap file question as

Swap file size (in decimal): [400] (from session)
New value:

If after you initialize the program file you run SPRED again on this file, it will display

Swap file size (in decimal): [400] (from file)
New value:

Before Running SPRED

To make informed decisions about the values you can change through SPRED, you need to know something about the target process and its relationship to other processes.

If you want to affect program loading and prepaging, you need to know about the form of data storage. For example, is the data storage static storage (in the .PR file) or is it on a runtime stack or heap?

Changing PID–size type is needed only if you want to run more than 255 processes on your system; it's needed to allow a process to run with a PID over 255. Changing PID–size type doesn't directly affect performance. To check a program's PID–size type, you needn't run SPRED directly; instead, use the macro PIDSIZE.CLI supplied with the operating system.

Changing program locality is needed only if you want to use process classes on your system and to base one or more classes on program locality. You might want to change program locality if you have purchased CLASP.

Generally, you should plan to work with only one program at a time. Then check performance under typical conditions, and repeat this change/check procedure until you see some improvement (or decide that using SPRED won't make any difference).

After you find SPRED settings that you like for a program, we suggest you create an argument file (and perhaps a log file) with them. These files will allow you to implement the settings easily in the future, whenever the program file is rebuilt.

SPRED is shipped in :UTIL, with a Help topic file. To get help, type HELP *SPRED and press NEW LINE.

To use SPRED, you need read/write access to the program file, read access to the program symbol table, and execute access to SPRED.PR. These are all the privileges needed to run SPRED as it is shipped.

SPRED doesn't check VSGEN parameters, so you should check them yourself before starting SPRED. If you are running AOS/VS, type TYPE :SYSGEN:sys.CSF and press NEW LINE to see which options the current the operating system allows. If you are running AOS/VS II, run VSGEN and list the current specification. Anything not enabled in VSGEN won't work. SPRED won't report an error, and the program will run as usual, but the action you specified with SPRED won't happen. To enable the SPRED option, you'll have to rerun VSGEN.

SPRED treats all addresses as octal and other numbers (like sizes) as decimal numbers of pages.

SPRED Menu Choices

When you start SPRED interactively (without the /ARGFILE switch) and give needed information, it displays the menu shown in Figure 11–10.

SPRED – SELECTIVE PREAMBLE EDITOR

Revision – n.nn.nn.nn
Filename – pathname
Symbol table – pathname

Do you want to:

1) Display settings from preamble of program file
2) Clear preamble of program file
3) Edit variable swap files
4) Edit program loading regions
5) Edit prepaging regions
6) Edit PID–size type
7) Edit program locality
8) Apply settings to preamble of program file
9) Create argument file

Type HELP for help, or BYE to exit from program
Enter choices, separated by commas:

Figure 11–10 The SPRED Menu

You select a choice by typing its number and pressing NEW LINE. You can specify multiple choices if you separate them with a comma. To get help, type HELP and press NEW LINE. You can exit by typing BYE and pressing NEW LINE. Some detail on the choices follows.

Option 1 "Display settings from preamble of program file" describes program loading, prepaging, swap file settings, PID–size type, program locality, and other information in the program preamble. If SPRED hasn't run on the program before, it displays the system default settings. The initial settings don't change until you initialize the new settings using option 8, "Apply settings to preamble of program file."

Option 2 "Clear preamble of program file" restores the original system default (0) to the program loading, prepaging, swap file, and program locality settings, and changes PID–size type to smallPID type. You can then either leave SPRED or specify other settings.

Option 3 "Edit variable swap files" allows you to change the variable swap file setting. The new setting isn't written to the program until you select option 8, "Apply settings to preamble of program file." Here's some background.

During memory contention, sometimes a process must be swapped to disk. If the process working set has grown while the process was running, the swap file might not be large enough to hold the working set. If this happens, the system must strip pages from the working set until it fits in the swap file. Later when the process swaps back in, it may need the stripped pages back and take many page faults to get them. These operations take a lot of time. You may be able shorten this time by setting up a swap file large enough to hold the working set. Working set sizes are displayed by the PED program, column WSS.

The default maximum swap file size is 126 pages. The default swap file size for all processes not given a custom size (with SPRED) is also 126 pages.

Neither allowing variable swap files nor specifying a large swap file will prevent or inhibit a program from running. The only negative aspect is the loss of disk space. Small swap files can cause problems: do not specify a swap file smaller than the process WSMIN or the number of pages it requires.

To check or change the swap file value, type 3 and press NEW LINE. Then type the new value (limit 512 pages). To retain the old setting (in brackets), press NEW LINE.

Option 4 "Edit program loading regions" allows you to specify pages to be loaded into memory when the process is created. The maximum number for noncontention and contention situations was specified at VSGEN.

Normally, when a process is created, its working set has only a few pages (nominally 0). Program loading multiple pages may be useful if the process will use many unshared pages. It's faster to load most (or all) of the needed pages when the process is created than to fault the pages in later.

Program loading is useful only for static storage — information that's kept in the program file. For example, in FORTRAN, a variable or array that's blank or named COMMON is static. Program loading is not useful for information kept on a runtime stack (FORTRAN) or heap (PL/I).

To check or change the program loading regions, type 4 and press NEW LINE. SPRED asks

Do you want to program load whole unshared space? [No]:

If you decide to program load, it's simple and effective to answer yes.

If you say no, you'll need to specify start and end addresses, which isn't easy for a high level language. Even a Link map gives no clear definition of program symbols and their addresses.

To load the entire unshared space, type Y and press NEW LINE. To specify addresses, simply press NEW LINE, and SPRED will ask for the Start address and End address.

Option 5 "Edit prepaging regions" allows you to specify that more than one page will be added to the working set on a page fault. Normally, when a process takes a page fault, the system adds only one page to its working set.

Having a process prepage can help when you know the process will often need more than one page when it takes a fault. (If it needs only one page, only one page will be brought in.) Prepaging can be useful for both static information (stored in the .PR file) and information stored on the runtime stack or heap. It's relevant for high level languages.

To check or change the prepaging setting, type 5 and press NEW LINE. SPRED then asks

Do you want to prepage the whole nonshared space? [No]:

This pertains to the part of the program in which you want prepaging to occur. It doesn't imply that the whole unshared space can be read on a fault. If you decide to prepage, we recommend a yes answer. A yes answer will enable prepaging to occur from the bottom of unshared space to the bottom of shared space. (If you answer NEW LINE, SPRED will ask for start and end addresses. It's hard to discover meaningful values for these.)

After you answer this question, SPRED asks

Prepage cluster size (in decimal):

The range of good answers is 2 (double the norm) to 8 (the size of page files). We suggest that you try a value of, say 2, run the program, and check performance. If you want, try another value and check performance again. To restore the original default, type 0 and press NEW LINE.

Option 6 "Edit PID–size type" allows you to tell the operating system to recognize this program as a smallPID, anyPID, or hybrid program.

When you select this option, SPRED asks

PID–size type: [default]

Choices are: (smallPID, hybrid, or anyPID) new value:

A smallPID program can run only in the PID range 1 through 255. It can't create a process with a PID above 255. (This means it can't create a process if 255 processes are running.) It can communicate with a process based on any PID–size type except possibly a process based on an anyPID program.

A hybrid program can run only in the PID range 1 through 255, but it can create a process with any PID. It can communicate with a process of any PID–size type.

An anyPID program can run in the entire PID range 1 through the maximum specified at VSGEN. It can create a process with any PID. It can communicate with a process based on any PID–size type except possibly a process based on a smallPID program.

Generally, you shouldn't change the PID–size type of a smallPID program with SPRED until the program has been checked for any smallPID restrictions. This check is relatively easy to perform; you can do it with the macro PIDCALL_CHECK.CLI shipped with the operating system.

You can change hybrid programs to anyPID programs without checking, as needed by your environment, if doing so won't jeopardize its communications with smallPID programs on your system. You can change an anyPID program to hybrid without checking it, as needed by your environment, if doing so won't prevent it from running. (A hybrid can't run if 255 processes are running.)

All these issues, and using macro PIDCALL_CHECK.CLI, are explained in more detail in Chapter 11, Fine–Tuning System Performance, in the section "Running More than 255 Processes on Your System."

Option 7 "Edit program locality" allows you to specify a program locality other than the default program locality.

A program's program locality helps determine the class in which the process will run. (The other determining factor is user locality, selectable by PREDITOR.) The default user and program localities are 0, which by default puts every process in the default class.

The main reason to select a nondefault program locality is to give a process a specific percentage of CPU (job processor) time. Other steps involve using the optional Class Assignment and Scheduling Package program (CLASP) to define a class including this program locality, allotting processor time to the class, and enabling class scheduling. For more information on classes and the optional CLASP program read Chapter 11, Fine–Tuning System Performance. Classes are explained fully in the manual *Using the Class Assignment and Scheduling Package*.

When you choose this option, SPRED asks

Program locality (in decimal): [default]

Type a number you want and press NEW LINE. The valid range is 0 through 15.

Option 8 "Apply settings to preamble of program file" tells SPRED to write the current settings to the program file overwriting the settings already there. You must do this if you want your efforts to be written to the program. Skip it when you're creating an argument file and don't want the values written, or when you want to cancel the session and start again.

When you type 8 and press NEW LINE, SPRED validates your answers syntactically: it doesn't check VSGEN limits or the program file. If the answers are okay, SPRED writes them to the program file. SPRED verifies by displaying the message

Settings applied successfully

Option 9 "Create argument file" allows you to create an argument file from the current settings. You can use the file later to edit a program without a SPRED session. When you type 9 and press NEW LINE, SPRED asks

Default argument file name: program–pathname_n.ARG

New name:

The default name is the program name (without .PR), plus _n (n is the sequence number, 1 for the first argument file for this program, 2 for the second, 3 for the third, and so on), plus .ARG. The default name for the first argument file for :UDD:BEN:TESTPROG.PR, would be :UDD:BEN:TESTPROG_1.ARG.

Take the default name or type a new one as desired.

When you're done with SPRED, type BYE and press NEW LINE to terminate it and return to the CLI.

SPRED Example

The example in Figure 11–11 shows a SPRED session for changing PID–size type.

```
) X SPRED TESTPROG.PR ↵

SPRED – SELECTIVE PREAMBLE EDITOR

Revision – n
Filename   – :UDD:JONATHAN:TESTPROG.PR
Symbol table – :UDD:JONATHAN:TESTPROG.ST

Do you want to:

        1) Display settings from preamble of program file
        2) Clear preamble of program file
        3) Edit variable swap files
        4) Edit program loading regions
        5) Edit prepaging regions
        6) Edit PID–size type
        7) Edit program locality
        8) Apply settings to preamble of program file
        9) Create argument file

Type HELP for help, or BYE to exit from program.

Enter choices, separated by commas: 6 ↵

PID–size type: [SMALLPID] (from program file)
Choices are: (SMALLPID, HYBRID, or ANYPID) new value: anypid ↵

... (header and menu display) ...

Enter choices, separated by commas: 8 ↵

PID–size type setting applied successfully

... (header and menu display) ...

Enter choices, separated by commas: BYE ↵

)
```

Figure 11–11 Sample SPRED Session

End of Chapter

Chapter 12
Submitting a Software Trouble Report (STR)

Read this chapter if software-based abnormal shutdowns occur and you want to submit a Software Trouble Report (STR) to Data General. For problems with hardware, run the diagnostics that come with your computer or contact your hardware support organization for assistance.

A Software Trouble Report is your way of informing Data General that you have found a serious error in AOS/VS or AOS/VS II, its utilities, or its documentation. The STR is also a way to request an enhancement or to ask a question. Filling out an STR takes time, so before you start consider the following:

- Some software problems result from disregarding requirements about which revisions of microcode and software you should run. See the Environment section of the current release or update notice for more information.
- Some STRs report problems Data General already knows about. Browse the Notes and Warnings sections of the current release or update notice to see if the problem has already been identified. The AOS/VS Monthly Newsletter also reports on problems, solutions, and workarounds. If you have a software support contract, contact your Customer Support Center to see if yours is a known problem.

This chapter reproduces pages from the on-line STR form, STR_FORM_AOSVSII, which is in :UTIL. (The AOS/VS version of this form, STR_FORM_AOSVS, is almost the same, but requires less information.) Before a software problem occurs, we suggest that you print copies of the STR form so that you will have one ready when needed.

This chapter does *not* help you diagnose a system problem. The manuals *Installing, Starting, and Stopping AOS/VS* and *Installing, Starting, and Stopping AOS/VS II* include a section, Emergency Procedures, that helps you diagnose and quickly recover from a hardware or software problem. Used in conjunction with those procedures, this chapter tells you which information to gather and what to send to Data General with your STR form to officially report such a problem.

The major sections in this chapter are:

- Filling Out the First Page
- Filling Out the Second Page
- Reporting System Panics or Hangs
- Reporting Process Traps, Terminal Services and MRC Controller Fatal Errors, and Call Tracebacks
- What to Send to Data General

AOS/VS II Software Trouble Report

Use this form to report one problem with AOS/VS II or its documentation. For help in filling out this form, see Chapter 12 in the manual "Managing AOS/VS and AOS/VS II" (093-000541). In North America, send the completed form to the address shown on the last page. Elsewhere, send it to your Data General representative. In North America, if your contract permits, you can call the Customer Support Center at 1-800-DG-HELPS. We suggest filling out as much of this form as you can beforehand. To print additional copies of this form, use the pathname :UTIL:STR_FORM_AOSVSII.

Your name: _____ Telephone number: _____

Company: _____ Today's date: _____

Address:_____

_____

_____

_____ Originator ID No.: _____

What kind of problem are you reporting:

___ Request for enhancement ___ Documentation error ___ Question

___ System/program failure ___ Software error ___ Other

Frequency:

___ Frequent ___ Occasional ___ Erratic ___ Reproducible

Severity:

___ Severe ___ Moderate ___ Not critical

What revision of AOS/VS II are you running? _____

What program or module do you think is causing the problem? _____

What is the problem? Please summarize below:

_____

_____

_____

NOTE: For documentation errors, please specify the manual title, its part number, and page number, or the Help file name or topic. You can photocopy the page in question.

If the problem is a documentation error or a question, you are finished filling out this form; otherwise, please continue.

Filling Out the First Page

First things first: **one problem per STR**. If you have two problems to report, file two STRs. You can file an STR for a problem with the operating system, one of its utilities, or a problem with your software that you think relates to the operating system. You can also file an STR about a problem with the documentation. Finally, you can submit an STR to ask a question or to request an enhancement.

If you have a question or want to bring to our attention a problem with the documentation, you need to fill out the first page only. Otherwise, you will need to fill out the first two pages, and possibly more.

Please follow these steps:

1. So that we may get in touch with you by mail or by phone, supply your name or the name of the person you want to be our contact person. You can give each STR an originator ID number so that you can track correspondence about your problem.

2. We sort out problems by type and importance. It saves us time if you identify the kind of problem you are reporting, its frequency, and its severity. Please only check "Reproducible" if you can also include a series of steps that always cause the problem to occur.

3. Please use the CLI SYSINFO command to get the revision of the operating system you are running.

4. Try to identify the program or module that is causing the problem. If you are not sure, give the name of the operating system.

5. In the three lines provided, summarize the problem or question. We use this summary to identify your problem in any correspondence. It will also be printed in the AOS/VS Monthly Newsletter. A good summary is one that lets us reproduce the problem or fully understand the enhancement you are requesting. (For documentation errors, you can photocopy the page in question and write the correction on the page.)

   Here's an example of an actual summary that did not require any elaboration:

   Using the block I/O system call ?RDB to an MCA with ?PRNH set to 0 directs AOS/VS II to accept transmissions from any MCAT. With VS rev. 4, after ?RDB returns, ?PRNH contains the original link. With AOS/VS II, ?PRNH contains 0.

If you are reporting documentation errors or have questions, you are finished filling out the form. If you are located in North America and your contract permits, you may give this information by phone to the North American Customer Support Center (call 1–800–DG–HELPS). If you are located in North America but do not have a support contract, send the form to:

Data General Corporation
1626 Jeurgens Court
Norcross, Georgia 30093

Attention: STR Administrator

If you are located outside of North America, contact your Data General representative for details on how to submit your STR.

AOS/VS II Software Trouble Report, Page 2

Other related software running on your system:

_____ Revision: _____

_____ Revision: _____

_____ Revision: _____

_____ Revision: _____

CPU Model: _____ Memory size: _____ Microcode Revision: _____

Peripherals: _____

_____

_____

Peripheral microcode revision: _____

Which patches have you applied? Are any of these special patches?

_____

_____

Is any software non-standard? _____

Are you using any non-DG hardware? _____

Have you made any recent changes to your system? _____

_____

_____

Give a detailed description of the problem below. Please include the date and time the problem occurred, what you think caused the problem, steps we can follow to reproduce the problem, and any other pertinent information. Continue on another sheet of paper if necessary.

_____

_____

_____

_____

_____

_____

_____

Filling Out the Second Page

Page 2 is more involved because it asks you to provide, in some detail, the context where the problem occurred. Each detail you provide helps us identify — or rule out — possible causes.

When filling out this page, please be specific. A problem with one revision of microcode may disappear in the next. It may be significant that a problem occurs on a particular date or at a particular time. As for questions about hardware, peripherals, and non-DG or nonstandard software, the more we know about the environment in which a problem occurs the easier it will be to solve the problem. Best of all is when you tell us exactly how to reproduce the problem.

Here's an example of a good summary followed by a good detailed description:

> In AOS/VS 7.68 when establishing a UVTA connection from a console, if a CHAR/RESET is issued, the connection will be terminated and a UVTA break file will be created with the error 31411 "Memory Deallocation Error".
>
> This can easily be reproduced by the following scenario:
> 1. Log onto a directly connected terminal (not a vcon ...) on a 7.68 machine.
> 2. Call [!host] and log on.
> 3. Char/reset will then create a break file and blow away the connection.

Accurate, concise, reproducible.

Some special cases require either more information and/or attachments:

Incorrect results — we may need a copy of the user program. If possible, include only the minimum number of statements necessary to demonstrate the problem. If the problem requires input data, be sure to send us some sample data. Fill out the last page of the form and send your STR and attachments to Data General.

System Panics or Hangs — we need the panic code data. See the next section, "Reporting System Panics or Hangs." In the case of system hangs, it may also help if you can identify a system event that should have happened but that did not. The reason for this is that at any one time the system may legitimately be in its idle loop. That is, the system may not be doing any work because it does not think there is any to be done.

Process Traps, Terminal Services Controller Fatal Errors, or Process Call Tracebacks — we need more data. See the section "Reporting Process Traps, Terminal Services and MRC Controller Fatal Errors, and Call Tracebacks," later in this chapter.

AOS/VS II Software Trouble Report, Page 3

For system panics or hangs, record the information below. After ESD copies the system dump file to file 0 of the tape, use the COPY command to copy the system symbol table file to file 1. Use the DUMP_II program to dump :ERROR_LOG to file 2. If you run SYSLOG, use REPORT to report on secondary errors and dump the report to file 3. For mid- and high-end ECLIPSE MV/Family systems (ECLIPSE MV30000 and above), see Chapter 12 of "Managing AOS/VS and AOS/VS II" for other information you should report.

AOS/VS II PANIC

Fatal Error Number: _____ JP value (if multi-processor): ___

Val1: _____ Val2: _____ Val3: _____ Val4: _____

Val5: _____ Val6: _____ Val7: _____ Val8: _____

Val9: _____ Val10:_____ Val11:_____ Val12 _____

AOS/VS II HANG

Type the break sequence, HALT <NL>, . <NL>, and CONTINUE <NL> three times. If PC values differ, repeat the sequence; we provide space for eight runs. If HALT prints status information, you may omit the . command. You may omit leading zeros when transcribing information.

	AC0	AC1	AC2	AC3	PC
First run:					
JP0:	_____	_____	_____	_____	_____
JP1:	_____	_____	_____	_____	_____
JP2:	_____	_____	_____	_____	_____
JP3:	_____	_____	_____	_____	_____
JP4:	_____	_____	_____	_____	_____
JP5:	_____	_____	_____	_____	_____
Second run:					
JP0:	_____	_____	_____	_____	_____
JP1:	_____	_____	_____	_____	_____
JP2:	_____	_____	_____	_____	_____
JP3:	_____	_____	_____	_____	_____
JP4:	_____	_____	_____	_____	_____
JP5:	_____	_____	_____	_____	_____

Reporting System Panics or Hangs

System panics automatically report a fatal error number on the system console, followed by a JP value (if more than one job processor) and twelve values.

System hangs require you to type the break sequence, the HALT command and the . command (on some CPUs, the HALT command incorporates the . command), and the CONTINUE command. You should repeat this sequence at least three times, but if PC values differ, repeat the sequence additional times. We know this is tedious, but it is the only way to isolate the code that may be the problem.

In either case, on a printing console, you can detach the printout and include it as an attachment rather than transcribe all of the data. If you have the MV/Data Center Manager product, you can print the logged information.

For both system panics and system hangs, you should run ESD as described in the *Installing* manual for your operating system. ESD will run and dump system memory to tape file 0 (this may require multiple tapes). (If you dump to a user–defined system area, a Data General representative could analyze the dump remotely and you would not need to copy the system symbol table or dump any other files to tape.)

In addition to capturing this information, you should also use the CLI COPY command to copy the system symbol table file to tape file 1 and use the DUMP_II program to dump the system error log to tape file 2. For AOS/VS II systems only, if you run system logging, use the REPORT program to report on secondary errors. Finally, if yours is a mid– or high–end ECLIPSE MV/Family system, there is other information you need to provide. The following sections show you what to do.

Copying the System Symbol Table File

With the memory dump tape mounted on unit 0, type SUPERUSER ON, make :SYSGEN your working directory, and then type the form of the COPY command that pertains to your type of tape drive:

```
COPY @MTx0:1 sys.ST                                  (MTC and MTJ drives)
or
COPY/ODENSITY=value @MTx0:1 sys.ST                   (MTB and MTD drives)
or
COPY/ODENSITY=value @MRCTAPE000A00:1 sys.ST          (MRCTAPE drives)
```

where

value — is the tape density. ESD always dumps to tape at the highest possible density: 1600 b/in for an MTB–type tape drive; 6250 b/in for an MTD–type or MRCTAPE tape drive. Do not specify tape density for MTC or MTJ drives.

x — is B, C, D, or J, depending on your type of tape unit (assuming your tape unit has a default device name).

sys — is the name of your tailored system, without the .PR suffix.

Dumping the Error Log

Make the root (:) your working directory and, using DUMP_II, dump to the next file on the tape, tape file 2, the system error log file, using the format

```
DUMP_II  @MTx0:2 ERROR_LOG
```

Reporting Secondary Errors (AOS/VS II with System Logging Only)

If you run system logging on your system, it helps us to have either a report of secondary errors contained in the system log file, or the system log file itself. (The system log file, :SYSLOG, can be very large, so a report of the secondary errors is preferable.)

To produce a report of secondary errors, type the command line

```
Su) X REPORT/SECONDARY_ERROR/L=SEC_ERRORS.OUT SYSLOG ↵
```

Then, use DUMP_II to dump the report to the next tape file, tape file 3. For example, type

```
Su) DUMP_II @MTJ0:3 SEC_ERRORS.OUT ↵
```

Or, dump :SYSLOG to tape file 3 by typing

```
Su) DUMP_II @MTJ0:3 SYSLOG ↵
```

Reporting Mid– and High–End ECLIPSE MV/Family System Problems

For mid– and high–end ECLIPSE MV/Family systems (ECLIPSE MV/30000 and above), there is an additional piece of hardcopy information: the revisions of all system hardware and microcode. And for ECLIPSE MV/40000 and MV/40000 HA systems, Data General also needs information from the SCP error log.

You can get a display of the revisions of all system hardware and microcode by using the SCP REVISION command. For example,

SCP–CLI/Jp0> REV ↵

Data General Corporation MV60000HA – Copyright 1993

SCP PROM revision	*0.34*
Diagnostics revisions	*21, 25, 18*
JP0 microcode revision	*37*
JP1 microcode revision	*37*

On an ECLIPSE MV/40000 or MV/40000 HA system, use the SCP VIEWLOG command with the /V switch to display the error log. Print or transcribe the error entries logged during the one hour preceding the system failure. For example,

SCP–CLI/Jp0> V/V ↵

TYPE DATE TIME MESSAGE
600 03/22/90 07:24:09 JP3 RESET Command Issued.

600 03/22/90 07:24:08 JP2 RESET Command Issued.

600 03/22/90 07:24:08 JP1 RESET Command Issued.

600 03/22/90 07:24:08 JP0 RESET Command Issued.

20 03/22/90 07:24:08 CP2 initialization complete

20 03/22/90 07:24:07 CP1 initialization complete

20 03/22/90 07:24:07 CP0 initialization complete

604 03/22/90 07:14:52 JP0 BOOT Command Issued.
Controller type: Eclipse Channel: 0(oct) Device: 24(oct)

[Type NL to Continue, CR to Exit]

... (Continue to display the events during the hour preceding the system failure) ...

AOS/VS II Software Trouble Report, Page 6

AOS/VS II Process Trap or User Process Trap

Name of the process: _____

Trap type: _____ PC: _____

AC 0: _____ AC 1: _____ AC 2: _____ AC 3: _____

Carry (C): _____

IMPORTANT: Use DUMP_II to dump the .BRK or .MDM file to tape.

Terminal Services and MRC Controller Fatal Errors

Use DUMP_II to dump the files in :TSDUMPS and the system .CONFIG file (in :SYSGEN) to tape.

If a process produces call traceback information:

Name of the process: _____

Error number: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

from fp: _____ pc: _____

Error message: _____

Reporting Process Traps, Terminal Services and MRC Controller Fatal Errors, and Call Tracebacks

This section discusses several error situations that require special treatment.

Process Traps

When a process traps, it will send a message to the user or system console and create a break file (suffix .BRK) or a memory dump file (suffix .MDM). For example, if EXEC fails, it will send a message to the system console, and it will create a memory dump file. Components of EXEC will also create .MDM files as well.

In all cases related to EXEC, please use DUMP_II to send the most recent .MDM files. These files, in :UTIL, are

.MDM file template:	For program file:
?EXEC+.MDM	EXEC.PR
XBAT+.MDM	XBAT.PR
XLPT+.MDM	XLPT.PR
XMNT+.MDM	XMNT.PR
XNET+.MDM	XNET.PR

In each of these cases, the + stands for the date and time the .MDM file was created.

Some break files or memory dump files will be located in :PER. You must dump these files before rebooting the system, since booting deletes and then recreates the :PER directory.

Use DUMP_II to dump the .BRK and .MDM files from :UTIL and :PER to tape.

Terminal Services and MRC Controller Fatal Errors

When a fatal error occurs in some MRC controllers and in most terminal controllers, AOS/VS II will send a message to the user or system console and copy the failed controller's memory to a disk file in the system directory :TSDUMPS.

A terminal services dump file has a name of the form

```
TS.dcode.n.ddmonyy.hhmmss
```

where

dcode	is the device code
n	is the processor number on multiple–processor controllers only
ddmonyy.hhmmss	is the time stamp assigned to the dump file

An MRC dump file has a name of the form

MRC.cc.nn.ddmonyy.hhmmss

where

MRC	stands for the MRC controller
cc	is the two–digit chassis number of MRC chassis where the controller resides, in hexadecimal notation
nn	is the node (slot) where the MRC controller resides, in hexadecimal notation
ddmonyy.hhmmss	is the time stamp assigned to the dump file

When submitting an STR about a failed controller, send the pertinent controller dump file on tape. Also include the system configuration file (sys.CONFIG, which is located in the :SYSGEN directory). You do not need to send a system memory dump or system symbol table file.

Call Tracebacks

Transcribe the name of the process, the error number, 14 lines of data, and the error message.

What to Send to Data General

As the last page of the form indicates, you can include supporting information along with the completed STR form.

```
                      AOS/VS II Software Trouble Report, Page 7

                      Attachments and Quantity

___ None           ___ Console output      ___ Listings  ___ Manual page(s)
___ Diskette       ___ Reel-to-reel tape   ___ Cartridge tape
___ Other (specify): ____________________________________________________

Please label attachments with your name, your company's name, the contents
and format of the media, the Originator ID No. (if you assigned one), and
"# of #" (e.g. "1 of 2").

In North America, send to:

                      Data General Corporation
                      1626 Jeurgens Court
                      Norcross, Georgia 30093

                      Attention: STR Administrator
```

End of Chapter

Chapter 13
Fine–Tuning System Performance

Read this chapter when you want to understand process and disk space concepts in order to run your system more effectively.

This chapter and the next one offer information and suggestions that can help the person who manages the system make sound system–oriented decisions. System security is discussed in Chapter 14.

Major sections in this chapter are

- Process Types and How to Use Them
- Running More than 255 Processes on Your System
- Multiple Processor Computers
- Classes and Logical Processors
- Disk Space and Performance
- Data Caching on AOS/VS II LDUs
- Using the HISTO and HISTOREPORT Utilities
- Using the LOGCALLS Utility

Process Types and How to Use Them

This section explains how AOS/VS and AOS/VS II manage processes; then offers some suggestions for your own system.

The operating system manages its resources quite efficiently, giving memory and CPU time to interactive and batch processes according to their types, priorities, and groups. There are some steps you can take to optimize system resources to meet your own needs.

As described in Chapter 1, AOS/VS and AOS/VS II are virtual memory, demand–paged systems. Virtual memory means that memory is a composite of physical (computer) memory and disk memory. Demand–paged means that the operating system adds a page to each process's working set of pages when the process requests it — or *on demand*. The operating system releases unused pages as processes require more memory. Some details on how the operating system handles memory management appear in the *AOS/VS System Concepts* manual.

Processes and Virtual Memory

The operating system runs each program as a process, with its own process ID (PID). A process starts with a certain number of pages of virtual memory usually from 5 to 100 pages. These pages are the process's initial working set. The working set is a subset of the process's total logical address space. When the process needs more pages (perhaps to execute a routine that isn't in memory), a page fault occurs. The operating system then adds an additional page to the process's working set. The theoretical limit on the number of pages in a process's working set exceeds 1,000,000 — providing a limit of 2 gigabytes on any process's logical address space.

There can be many processes running simultaneously. Memory contention occurs when all currently active processes (including the operating system and its peripheral manager) require a working set larger than the computer's physical memory. Memory contention can occur much of the time.

In light memory contention, the operating system resolves the situation by removing inactive pages from processes and storing their images in its PAGE directory. The processes remain in physical memory, but with fewer pages in their working sets. Later, if demanded, the pages are restored to the working sets. This is called *paging*.

In heavy memory contention, the operating system removes whole processes (selecting blocked processes first) and stores their images in its SWAP directory. Their entire working sets are removed from memory. Later, the operating system restores the working sets to physical memory and the processes can run again. This is called *swapping*.

If there is no memory contention, no paging or swapping occurs. If there is contention, processes may be paged or swapped to disk on the basis of their process priorities.

Getting control of the processor (CPU) is a two–phase operation. Before a process can do it, its working set must be in physical memory.

Processes and Physical Memory

Processes are given physical memory according to type; priority is secondary. The process types are

- resident
- pre–emptible
- swappable

Table 13–1 shows how the the operating system scheduler allots each process type memory.

Table 13–1 Process Type Memory Allocation

Process Type	How It Gets Physical Memory
Resident	Gets memory on demand and keeps it. The scheduler retains the process's initial working set in memory; the system can page the process but can not swap it.
Pre–emptible	Gets memory if the memory is not needed by a resident process. The scheduler swaps a pre–emptible process if • A resident or higher priority pre–emptible process requires memory; or • The process becomes blocked and any other process requires memory.
Swappable	Gets memory if the memory is not required by a resident or unblocked pre–emptible process. The scheduler can swap the process as needed during memory contention.

Any process may have pages stolen from it by any higher priority process — or by the operating system itself — regardless of its type.

By default, all user processes, including batch processes created for users, are swappable.

Processes and CPU Time

The operating system scheduler determines how much CPU time a process receives. With AOS/VS Revision 7.00 and AOS/VS II Revision 1.00 and later, Data General provides class scheduling, and with some computers, multiple processors. (In a multiprocessor computer, each CPU is called a job processor.) All computers use standard scheduling and can use class scheduling as well.

Standard scheduling, which forms the basis for all scheduling, works like this.

The operating system scheduler runs on each job processor. When a job processor becomes free, that scheduler runs the highest priority ready process. The process can use the processor for a subslice period (32 milliseconds) if it wants.

If the process reaches the end of a subslice, or if a higher priority process becomes ready, the operating system reschedules. Again, the scheduler runs the highest priority ready process. The chosen process can be the same process or a different one.

There are two methods of scheduling: round robin and heuristic. With round–robin scheduling, the scheduler tries to give each process at the same priority an equal chance to run. A process that uses a lot of processor time isn't penalized.

With heuristic scheduling, the system may reduce a process's internal priority, based on process behavior. A process that uses a lot of processor time is penalized — relative to other, more interactive processes at the same priority.

The kind of scheduling a process gets depends on its group (explained later in this chapter). By default, user processes (and their sons) start in group 2, the heuristic group.

Standard scheduling is meant for general purpose systems. Since by default user processes are scheduled heuristically, highly interactive time–sharing processes are favored over compute–bound processes. This interactive bias can be overcome, as desired, if you give user processes a priority that makes them group 1 or group 3. Standard scheduling allows high priority processes (like the PMGR) to get all the time they need without penalty.

Also, compute–bound processes of the same priority tend to get equal amounts of processor time, regardless of group. This allocation of resources works well in situations where you want to treat compute–bound processes equally.

However, standard scheduling, based on priority only, does have two limitations, as follows:

- You can't give one compute–bound process priority over another compute–bound process without starving the lower priority process.
- You can't give compute–bound processes lower priority than interactive ones without the risk that the compute–bound processes will get no time.

With classes and logical processors, you can specify the percentage of time for processes in a class — overcoming both limitations. See the sections "Multiple Processor Computers" and "Classes and Logical Processors" later in this chapter.

Whether you use classes or not, ready processes get CPU time entirely on the basis of priority within process group. The next section explains how.

Priority Groups

There are three priority groups, whose ranges are established at VSGEN. They are

Group 1 (high priority, default VSGEN range 1 through 255);

Group 2 (medium priority, used for user processes, default VSGEN range 256 through 258);

Group 3 (low priority, default VSGEN range 259 through 511).

- Group 1 (high priority) processes include resident or pre–emptible processes with priority 1, 2, or 3. Group 1 processes include any processes with priority 4 though n. The n is the VSGEN "lowest priority for group 1" choice, default is 255. Group 1 processes of equal priority get CPU time on a round–robin basis: the system doesn't watch or predict their behavior.

- Group 2 (medium priority) processes include all processes whose priority ranges from n+1 through m. The n, as above, is the VSGEN "lowest priority for group 1"; m is the VSGEN "lowest priority for group 2." The VSGEN default range is priority 256 through 258.

  Group 2 processes also include any swappable processes with a displayed priority of 1, 2, or 3. By default, this includes user processes and all sons of the master CLI. The system shows swappable priorities n+1 through n+3 as 1, 2, and 3 for compatibility with early AOS/VS revisions before 4.00.

  Group 2 processes are scheduled heuristically, according to their behavior. Based on a group 2 process's prior behavior, the system expects it to have a given number of blocking events within a given time. If the process has more blocking events than expected in this given time, the system raises its internal priority, which gives a better shot at the CPU than other equal–priority group 2 processes. On the other hand, if a process has fewer blocking events than expected within a given time, it is starting to monopolize the CPU. The system then lowers its internal priority, favoring other equal–priority group 2 processes. Thus, within group 2, both priority and the number of blocking events determine which process next gets control of the CPU.

  In Group 2 scheduling, interactive processes (which tend to have many blocking events, where people pause for thought), are favored over more CPU–intensive (noninteractive) processes. Thus, group 2 scheduling is ideal for user processes that should each get about the same amount of CPU time. It is not ideal for processes that can use a lot of CPU time without blocking — processes like large sorts and batch jobs.

- Group 3 (low priority) processes are any type of process with priority m+1 through 511. The VSGEN default range is 259 through 511. Unlike group 2 processes, group 3 processes of equal priority get CPU time on a round–robin basis: the system doesn't care about their behavior.

  Within groups, the absolute difference between priority numbers has no effect — only the relative difference. This means that running five processes at resident priority 1 and priorities 4, 18, 200, and 201 is exactly the same as running them at resident priority 3 and priorities 4, 5, 6, and 7 if they are the only processes on the system.

Choosing Process Types, Priorities, and Groups

Process types, priorities, and groups offer a lot of flexibility, but things will be simpler if you keep the defaults where possible.

If you take no action to control process types or priorities, nearly all processes will be swappable, priority 2, group 2. This is the master CLI, EXEC, and PREDITOR default type and priority. The qualities and tradeoffs on the types are as follows:

- Resident Processes. A resident process cannot become blocked. Its working set always remains in physical memory. A resident process can wire additional pages into its working set with the ?WIRE system call. These pages cannot be paged out; they will stay resident, part of the working set, until the process issues an ?UNWIRE call or terminates.

  The operating system always runs its peripheral manager as a resident process, priority 2. If you have synchronous lines, you must run the GSMGR process resident, as shown in the *Installing* manual. Other Data General products are meant to run resident, as suggested in the product documentation; you should follow the suggestions. Any process that defines a user device (?IDEF system call) must be resident and must wire all device–handling pages so it can service device interrupts.

  Generally, try to avoid running a process resident unless you must, since the memory for at least its working set minimum is lost to other processes while it runs. If a process must be resident, run it at a lower priority than the PMGR — perhaps 2, 3, or even 511.

- Pre–emptible Processes. A pre–emptible process stays in physical memory as long as it remains unblocked, unless a resident or higher priority pre–emptible process needs the memory. This is a good general–purpose type for a process you want to favor over swappable processes.

  Processes that you might consider making pre–emptible are the master CLI, PID 2. You should also give it a high priority. Giving the master CLI this advantage can help it get CPU time to control the system if another process runs wild. It consumes few (unshared) pages. Another possibility, if you want fast print request handling, is EXEC's XLPT co–operative.

  Other candidates include any important short–term processes. For example, assume a short–term process needs to run uninterrupted every 10 minutes, and will then issue a delay call. You might make it pre–emptible, with the highest priority (1). Then, the scheduler will not permit any lower priority process to displace it while it runs. When it issues the delay call, it becomes blocked and the operating system may swap it out. When the delay period has expired and the process is ready again, it can displace any other lower priority nonresident process in physical memory.

- Swappable Processes. Swappable, priority 2 (actually VSGEN n+2, default 257), is the default for all user processes and their sons. These are group 2 processes. They all compete on an equal footing, and the scheduler tries to give them equal CPU time. Generally, you should take the PREDITOR and system defaults, which makes each process a swappable group 2.

If a swappable process can use a lot of CPU time productively, you can create it with (or change it to) a group 1 priority (like 4); this gives it round–robin scheduling (to avoid penalties for not blocking often), gives it priority over standard user processes, and yet maintains its swappable type.

To give a swappable process a slight advantage, you can give it higher priority, like 1, within the swappable range. This is often done with EXEC and/or XLPT. EXEC manages queues and XLPT manages printers; thus they can benefit from the higher priority.

You can't really decide on the best types, priorities, and groups for your processes until you've had some experience with your applications software. So we suggest that you run with the defaults for a while before setting up your own process parameters. Table 13–2, next, shows the relationships between type, priority, and group.

Table 13–2 Process Types, Priorities, and Groups

Process Type	Priority (Using VSGEN Defaults)	Group and Type of Scheduling
Resident or Preemptible	Priority 1 Priority 2 Priority 3 Priority 4 . . . Priority 255	Group 1, Round–robin
Any	Priority 1 (actual 256) (swappable) Priority 2 (actual 257) Priority 3 (actual 258)	Group 2, Heuristic
	Priority 259 Priority 260 . . . Priority 511	Group 3, Round–robin

Priority Cautions

Generally, don't allow a user process to run at resident or pre–emptible priority 1 or 2. (A possible exception is a user process that defines a device via call ?IDEF.) Also, *never* make EXEC resident or pre–emptible with a priority of 1 or 2. Having EXEC or a user process run at resident or pre–emptible priority 1 or 2 may create a problem with the operating system peripheral manager (which is resident at priority 2). Running EXEC as a resident or pre–emptible 1 or 2 can create a system deadlock forcing you to shut down and restart the operating system.

To maintain fast log–on service and other EXEC functions, allow EXEC to run at a higher priority than most user processes.

By default, user processes are created at swappable priority 2 (actual priority 257). You can prevent users from changing priority when you create user profiles by taking the default answer No to the PREDITOR question *Change priority?* By default, EXEC is created with swappable priority 1 (actual priority 256) and you needn't change this.

If you do decide to run EXEC pre–emptible or resident, *change* its priority to 4 (or a higher number) before changing EXEC's process type.

Page Faults and Program Design

Good program design is at least as important as the process types and priorities you choose for your system. A very important goal here is minimizing the number of page faults.

When a program takes a page fault, the operating system must add one or more memory pages to its working set often via a disk access.

If a little–used program takes many page faults, this is unimportant. But if often–used programs (like your main application programs) take a large number of page faults, everyday system performance will suffer.

Page faults are directly related to program design. Programmers can design programs to localize references, which will reduce page faults. To help with this program design, they can use a Data General programming tool like HISTO, which can tell where a program is spending most of its CPU time and show where references should be localized, and LOGCALLS, which shows what a program is doing, and how often.

For often–used programs with many unshared pages, you may be able to improve performance by program loading multiple pages and/or adding more than one page to the working set on a fault. To do either of these, use the SPRED utility (Chapter 11).

Programmers can make use of shared pages. Many Data General products, like the CLI and the SED text editor, do this. A shared page can be used by more than one process. Shared pages conserve physical memory and reduce faults, since the operating system does not page a shared page out to disk immediately when a process releases it. Shared page concepts are given in the *AOS/VS System Concepts* manual; they can be applied to higher level languages in addition to assembly language.

Basically, fine tuning your processes requires some planning and experience. With your typical applications up, run PED and watch the FTA (page fault) and WSS (working set) columns. Listen to user comments, and note the amount of time programs like big sorts and/or batch jobs take. When you change a process–related setting, note the effects — user response time, FTA figures, total performance — and decide whether or not changing the parameter helped. You might want to change settings based on the time of day.

Ideally, there will be a balance between the resources that interactive (I/O–bound) and noninteractive (compute–bound) processes use. The whole approach depends on what kind of operations you want the system to perform at any given time.

Creating Different Kinds of Processes

The primary process–creating command is PROCESS. It creates a son process with the type, priority, and other parameters specified with switches.

A process can display or change another process's (or its own) type with the command PRTYPE. It can display or change priority with the command PRIORITY. To learn a process's group, use Table 13–3.

Table 13–3 How Process Groups Relate to Type and Priority

Type	Priority			
	1 – 3	4 – n (default is 4 – 255)	n+1 – m (default is 256 – 258)	m+1 – 511 (default is 259 – 511)
Resident or pre–emptible	Group 1	Group 1	Group 2	Group 3
Swappable	Group 2	Group 1	Group 2	Group 3

No process can create a son with, or give itself, privileges that are not granted in its user's user profile. The PREDITOR defaults don't allow a process to change priority or type.

The master CLI and any of its CLI sons (except LOCK_CLI and a locked 32–bit CLI) can change its type or priority; it can create sons of different types and priorities.

So the system operator, via a privileged CLI, can always create any kind of process; for example,

```
) PROCESS/BLOCK/IOC/RESIDENT/PRIORITY=1/SUPERPROCESS PROG ↵
```

Probably you will not want to create these processes directly. Instead, after deciding on your process parameters, you will want to put them in the UP.CLI macro.

If you want users to be able to create processes of different types and priorities, you must edit their profiles to allow this. Do it with caution because misuse of these privileges can allow undesirable processes to dominate the system.

Table 13–4 names the PREDITOR profile privileges in the order asked, and describes the effect on the user's processes. Each PREDITOR question is discussed in more detail in Chapter 2.

Table 13–4 Profile Privileges that Relate to Process Control

Privilege	What It Allows
Create without block	The user process can create sons (that in turn can create sons) without blocking. This allows the user process to proliferate. All its sons (and grandsons) can remain unblocked, increasing system overhead. This privilege relates to the next two. A CEO user needs this privilege.
Unlimited sons	These two parameters control the number of sons the user process can create. Each son can have all the privileges of its father (if desired). Generally, a user should not have the unlimited sons privilege. Each son requires some overhead, especially if its father isn't blocked (first privilege, above).
Change priority	The user process (and sons) can change its own priority if it has this privilege.
Change type	The user process (and sons) can change its own type (e.g., to resident) if it has this privilege. Do not give users this privilege if you want a secure system.
Change username	The user process can create son processes with different usernames. Effectively, this allows the user to become another user on the system, with OWARE access to that user's files. A user who can change his or her username to OP, for example, could issue EXEC commands. Do not give this privilege to users if you want a secure system.

(continued)

Table 13–4 Profile Privileges that Relate to Process Control

Privilege	What It Allows
Access devices	The user process can define a new device (call ?IDEF), wire pages, access the device, and/or remove the definition. Also, for this privilege to work, the process must be resident, which means the user must also have the "Change type" privilege. These privileges give the user at least the power to dominate or bring down the system.
Superuser	Of itself, this privilege doesn't affect the multiprocess environment, but superusers can run PREDITOR and give themselves any privilege. Do not give users this privilege if you want a secure system.
Superprocess	The user process (and sons) can change its own type and priority; it can block or terminate any other process. Do not give users this privilege if you want a secure system.
System Manager privilege	This privilege allows the user to initialize and release job processors (if your computer has more than one job processor), create and delete classes and logical processors (usually done via the optional CLASP utility), and change the locality of other users' processes. System Manager privilege also allows a user process to issue operating system calls that change the system date, time, ID (SYSID), and bias factor. Also, the user can start or stop the system log (SYSLOG). These privileges have significant impact on security. Use of classes and privileged system calls can affect the performance and security of your system. Generally, the master CLI issues all commands that require System Manager privilege; do not give it to any other user unless he or she really needs it.

(continued)

Table 13–4 Profile Privileges that Relate to Process Control

Privilege	What It Allows
Change working set limit	The user process (and sons) can override the working set limit that the operating system dynamically assigns to each process. The process must have this privilege to exercise any of the four working set privileges below: Priority, Logical address space, Minimum working set size, and Use other localities.
Priority	If given a higher than default priority, the user process (and sons) will get CPU preference over processes with lower priority.
Logical address space – batch	These parameters give the user process (and sons) a nondefault number of memory pages in batch or interactive processing. The default is a very high number. Giving specifics might have some effect.
Logical address space – non–batch Minimum working set size – batch Maximum working set size – batch Minimum working set size – non–batch Maximum working set size – non–batch	These parameters give the user process and sons specific minimum and maximum working set limits in batch or interactive situations. Giving specifics can have significant effect (usually bad) depending on your applications environment.
Use other localities – non–batch Use other localities – batch	Changing the user locality by itself confers no special powers, but if you run your system with classes (described later in this chapter a user might be able to run programs faster. Be careful to whom you give this power.

(concluded)

EXEC Process Control Options

EXEC has a number of commands to tailor the multiprocess environment (described in Chapter 3). They are

LIMIT	Can limit CPU time for batch streams or page usage for line printers.
PRIORITY	Can change the process type and priority of batch and XLPT printer processes (combines CLI commands PRTYPE and PRIORITY for these).
QPRIORITY	Can direct batch streams or devices to accept only requests that fall within a given queue priority range.

PID–Size Types

When you execute a program, it runs as a process. The PID–size type of the program determines the PID–size type of the process.

Table 13–5, next, summarizes program and process PID–size types; Table 13–6, which follows, gives details on program and PID–size types.

Table 13–5 Program and Process PID–Size Summary

Program PID–size Type (stored in program file)	Action	Process PID–size Type (stored in process runtime packet)
SmallPID	XEQ PROCESS command	A
Hybrid	XEQ PROCESS command	B
AnyPID	XEQ PROCESS command	C*
* — If a PID above 255 is available; otherwise, B.		

Table 13–6 Program and Process PID–Size Types

Program PID–Size Type	Process PID–Size Type
SmallPID–type program. A smallPID program can't run if PIDs 1–255 are in use. SmallPID is the PID–size type of all programs before AOS/VS Revision 7.00 (except the CLI and EXEC, which were hybrid in Revision 6.00). The Link program creates programs of smallPID size by default. A smallPID–type program will always be a type A process.	**A–type process.** An A–type process has a PID between 1 and 255. It can't execute any program if PIDs 1–255 are in use. Error conditions may result if a process with a PID over 255 tries to communicate with an A–type process. A is the PID–size type of all pre–AOS/VS Rev. 7.00 processes (except CLI and and EXEC; also CEO_CP, which is anyPID).
Hybrid program. Someone has labeled this a hybrid type program by linking it with the Link switch /PID_SIZE=HYBRID or by editing the program file with the SPRED editor. (You can tell SPRED to label "any" program as hybrid, but if the program has any small–PID limitations, the process may not be able to communicate with PIDs above 255. And, since the operating system thinks the process is a legitimate hybrid, it won't detect any PID–range errors. A hybrid program can't run if PIDs 1–255 are in use. Most programs shipped with AOS/VS and AOS/VS II are hybrid programs.	**B–type process.** A B–type process has a PID between 1 and 255. It can't run if PIDs 1–255 are in use, but it can create and and communicate with a process any PID–size type. Most Data General programs, including the CLI and EXEC, run as B–type processes. By default, each user's CLI is a B–type process. Thus, by default, a user CLI must run in the range 1–255 but can execute any PID–type program. Most processes from programs supplied with AOS/VS and AOS/VS II are B–type processes.

(continued)

Table 13–6 Program and Process PID–Size Types

Program PID–Size Type	Process PID–Size Type
AnyPID program Someone has labeled this a hybrid type program by linking it with the Link switch /PID_SIZE=ANYPID or by editing the program file with the SPRED editor. (As with a hybrid program, you can tell SPRED to label any program as anyPID, but if the program has smallPID limitations, communication errors may occur and go undetected by the operating system.) An anyPID program can run at any PID up to the maximum specified at VSGEN. The system will run it above 255 if possible.	**C–type process**, if a PID above 255 is free. **B–type process** if no PID above 255 is free. A C–type process can execute any PID–size type programs. But error conditions may arise after it executes a smallPID program (since the father process has a PID the son can't understand).

(concluded)

Because an anyPID program can run at any PID, it's the most flexible choice. Its only disadvantage appears above PID 255, where A–type processes may not be able to communicate with it. You can avoid this disadvantage by making all smallPID programs hybrid or anyPID.

The PID–size type arrangement shipped with the operating system provides maximum versatility. It allows you to specify more than 255 PIDs during VSGEN, and run any PID–size type program from user CLIs without errors.

Running More Than 255 Processes on Your System

By default, the operating system can run as many as 255 processes. With most computers, 255 processes is enough — it allows at least 60 people to use the system simultaneously.

(As a general rule, a CLI user who will not use the Data General CEO system needs an average of three processes — the CLI and a text editor, compiler, Link, SWAT debugger, or BASIC process. A CLI user who may also use the CEO system needs an average of three and a half processes — the CLI and subordinate processes like BASIC or CEO control program process and CEO Word Processor process. (Two word processors will increase the average number by 1.) A CEO–only user needs an average of two and a half processes. CEO–only users chain from the CLI, thus they don't need a CLI; they need only the two CEO processes. In any case, the half process covers extra processes like the SWAT debugger, CEO Spelling, or CEO Spreadsheet.)

So, assuming 20 or so processes for the system, 255 processes is enough to support 65 CLI users or 90 CEO–only users simultaneously. If you think that 255 processes is enough — true in the majority of cases — skip to the next section "Multiple Processor Computers." On the other hand, with a computer like an ECLIPSE MV/20000 or MV/10000, you may well want to run more than 255 processes. To do this, read on.

First, some definitions. Any PID over 255 is called a big PID. Any program not limited to communication with PIDs below 256 is compatible with big PIDs. (Limits are based on certain system calls whose PID field can hold only numbers 1 through 255. These calls are ?PSTAT, ?IREC, and ?EXEC.)

You can identify a process as big–PID compatible via a quick edit of its program file. The operating system will then accept the process as compatible with big PIDs.

In terms of resources, processes with big PIDs are identical to other processes. They require the same amount of system overhead, and they are governed by the same process type (swappable, pre–emptible, resident) and priority rules as any process. As before, system response generally relates to the number and type of processes that are running.

Creating a System for Big PIDs

To create a system that can run big PIDs productively, you must use the following steps:

1. Select a nondefault choice for the maximum number of processes at VSGEN. This enables the the operating system system to run PIDs above 255. There's little extra overhead involved in allowing as many as — say — 500 or 600 processes. The maximum number of processes allowed by AOS/VS VSGEN is 1,024. AOS/VS II VSGEN allows a maximum of 4,095 processes.

2. Ensure that any program not created by Data General that may run above PID 255 or communicate with a PID above 255 doesn't do anything that limits it to small PIDs. These limiting factors are

   - Checking PIDs of son processes with system call ?PSTAT.
   - Checking process termination or obituary messages with call ?IREC.
   - Asking service from EXEC with call ?EXEC, function ?XFSTS. (This function gets the process's relationship to EXEC).

   If a program doesn't do any of the things above, and it has no functional restriction to small PIDs (for example, such a restriction would be checking PIDs using a one–byte variable), then it's not limited to small PIDs. It's big–PID compatible.

   An easy way to check a program for small–PID limitations is to run the DG–supplied macro PIDCALL_CHECK.CLI on it. This macro searches a file for potentially significant system calls. If the macro finds none of the calls, then the program has no structural small–PID limits. (The macro may find call ?DADID, but ?DADID is okay if there are no other small–PID limits.) Unless the program has a structural restriction, it can handle big PIDs without requiring changes to the source code. The PIDCALL_CHECK macro also checks object files, overlay files, and libraries.

   Generally, if a program is written in a compiled language (FORTRAN, COBOL, C, PASCAL, PL/I), and it doesn't use the system calls mentioned above directly (by a high–level language system call function), then you need not change its sources to handle big PIDs.

   Usually, you don't even need to recompile and relink compiled–language programs to handle big PIDs. You must recompile and relink only if your old compiler produces code limited to small PIDs.

   Checking and configuring programs for compatibility with any PID is explained later, in "Checking and Configuring Programs for Any PID Size."

3. Tell the operating system to recognize each program you checked or configured in step 2 as compatible with big PIDs. To do this, use the SPRED editor to change the program PID–size type.

These steps enable a system to run PIDs larger than 255, maintaining the ability of processes to communicate. The steps don't necessarily produce PIDs above 255, nor do they prevent a program with small–PID limitations from running.

Most Data General utility programs supplied with the operating system, including the CLI and EXEC, can communicate with any process. DG has executed steps 2 and 3 on them. By default, every initial user process is the CLI. Thus, by default, any user CLI process can communicate with a process that has any PID, above or below 255. As before, user CLIs run with a PID of 255 or below.

Checking Program and Process PID–Size Type

You can check a program's PID–size type via SPRED and SPRED's PID–size choice, or more conveniently, you can use the macro PIDSIZE.CLI, supplied with the operating system. This macro runs SPRED for you and gets and displays the PID–size type. It takes about 10 seconds to run. For example,

```
) PIDSIZE :CLI.PR ↵
... (pause) ...

PID–size type [HYBRID] (from program file)
)
```

This shows that the CLI is a hybrid program.

To check a process's PID–size type, run PED with the /PIDSIZE switch. You can create a macro to show this and most other PED information by writing the following command line into the macro file (call the macro, say, PP_SIZE.CLI):

```
PED/CYCLE=10/PID/USER/PROGRAM/ELAPSED/CPU/BS/PIDSIZE/IO/FTA/WSS
```

A typical PED display, showing the 32–bit CLI as a B–type process (as Data General ships it) and the CEO Control Program as a C–type process, might look like Figure 13–1.

PID	USERNAME	PROGRAM	ELAPS	CPU	BS	PSZ	I/O	FTA	WSS
1	PMGR	PMGR	2/57	2:12		A	1836	262	271
9	OP	SMAP	2/42	38.11	B	B	885	1082	239
11	OP	XTS	2/42	5.89		B	313	456	466
14	OP	DCS	2/41	2:02		B	1153	1161	713
23	OP	MTA	2/40	8.82	B	B	8186	348	358
26	CEO_MGR	CEO_NSA_NO	2/37	0.73	B	B	93	553	349
29	CEO_MGR	CEO_POA	2/37	1.68	B	B	5175	192	199
37	CEO_MGR	CEO_GPN	2/36	0.30	B	B	10	262	195
45	STEVENS	CLI32	2/26	16:08		B	6832	4711K	312
46	STEVENS	.EMACS.	1/11	3:13	B	A	3509	201	208
67	JONATHAN	PED	12.00	0.15		B	0	147	154
263	JRH	CEO_CP	1/05	10.36	B	C	2689	834	515
266	PILAT	CEO_CP	1/41	1.89	B	C	813	389	375

Figure 13–1 PED Display with Big PIDs

The PED display shows most processes as B–type processes, running below PID 256. The two CEO Control Program processes — for users JRH and PILAT — are C–type processes, running above 255.

To some extent, you can tell the program PID–size type from the process PID–size type, as follows. An A–type process can come only from a smallPID program. A B–type process can spring from either a hybrid or anyPID program; PED can't tell you which. A C–type process can come only from an anyPID program.

Checking and Configuring Programs for Any PID Size

This section tells how to check your programs for small–PID limitations, and eliminate these if you find any. After ensuring that any program you may want to use in the big–PID environment has no such limitations, you can change the program PID–size types at will.

Checking Compiled Language Programs

In a program written in a high–level language like C, PASCAL, FORTRAN, COBOL, or PL/I, run the macro PIDCALL_CHECK.CLI on the program file. This macro checks for the occurrence of any of the system calls that may limit a program to small PIDs.

Macro PIDCALL_CHECK.CLI works with both 32–bit and 16–bit program files, and with object (.OB) files and libraries. If you use it to check a 16–bit program file, be sure to run it also on the overlay file (programname.OL) if there is an overlay file. The macro uses the FED text editor to search for the call code, in both 32–bit and 16–bit form. It takes 30 to 60 seconds to run, and produces a report on the pertinent calls. For example,

```
) PIDCALL_CHECK :UDD:DATABASE:ITEMIZE.PR ↵
... (pause) ...

Analyzing :UDD:DATABASE:ITEMIZE.PR — please wait 30 to 60 seconds.

... (another longer pause, then the macro displays the text of the report) ...

The report on the PID–size calls is in file
:UDD:JACK:ITEMIZE.PR.BP.REPORT
)
```

The macro writes its report to directory :UDD:username, from which you can print and file it. Only read access to the target file is required. If the macro finds no occurrences of call ?PSTAT, ?IREC, or ?EXEC, you're nearly done. There are just a few other things to check.

It's possible for one of the limiting calls to occur at runtime without being built into the program code. A limiting call can occur at runtime if the process issues a system call function with a runtime variable as the system call value. (If a program issues a system call function with a constant as the system call value, the call will be present in the program file, and the macro will find it.) Thus, you may want to have someone check the program sources for the appropriate system call text string; then see if a variable that could possibly evaluate to 5, 22, or 216 follows the call text. If there is such a variable, you need to follow the steps described in the next section. (Values 5, 22, and 216 are the values for system calls ?PSTAT, ?IREC, and ?EXEC.) The system call text strings are

C	sys or sys_ (not system) (case sensitive)
COBOL	CBSYS (case sensitive)
FORTRAN 77	ISYS (case insensitive)
Pascal	SYS (case insensitive)
PL/I	SYS (case sensitive)

If the macro finds no occurrences of calls ?PSTAT, ?IREC, or ?EXEC, and there is no system call function based on a runtime variable, only structural limits remain. (For example, the program reads PIDs into a one–byte variable, limiting the number to 256.) Structural limits like this may exist, and you should ask your programming staff to see if there are any. If there are structural limits to PID numbers in the program, the source code must be changed to eliminate them; then someone must recompile and relink the program.

If none of these checks reveals a limitation, then the program has no inherent smallPID limitation. You can use SPRED to change its PID–size type to hybrid or anyPID at will. Generally, if the program won't need to communicate with a smallPID program, choose anyPID, since anyPID can run at any PID.

Checking Compilers

Most Data General compilers of recent revision have been checked for generation of big–PID compatible code. Unless someone has used a system call function to call ?PSTAT, ?IREC, or ?EXEC in an application program, you can change application programs created by recent compilers to hybrid or anyPID type at will.

A compiler itself may not run above PID 255. If it cannot, anyone who tries to run it when PIDs 1–255 are in use will receive the error message

Too many processes

Arrange to have users run such a compiler only when PIDs below 256 are free.

Only the most recent versions of the SWAT debugger can handle PIDs over 255. To use SWAT to debug a program that will run above PID 255, you must relink the program with the newest SWAT loaded, using the /DEBUG switch.

If you can't tell whether a compiler generates code with small–PID restrictions, run macro PIDCALL_CHECK on each application program file.

If the macro doesn't find call ?PSTAT, ?IREC, or ?EXEC, the program has no small–PID restrictions. (Very few application program files contain these calls.)

After you determine that a program has no small–PID limitations, or after you eliminate such limitations, you're all set. You can run SPRED on the program file and label it a hybrid or anyPID program, as described in the section "Changing PID–Size Type."

If the Macro Reports System Call ?PSTAT, ?IREC, or ?EXEC

If the PIDCALL_CHECK macro reports system call ?PSTAT, ?IREC, or ?EXEC, have a programmer check for direct calls to the system (via the system call function for the language) in the program sources. If the macro reports only ?DADID, this is not a problem, as explained in the next section.

If there are direct calls to the system, talk with your programming staff and verify that the system calls (?PSTAT, ?IREC, and/or ?EXEC) have no small–PID limitations. These limitations are described later, under "Limiting Contexts in Calls ?PSTAT, ?IREC, and ?EXEC." If there are small–PID limitations, arrange for the sources to be changed — if possible — for big–PID compatibility. After the change, have the program recompiled and relinked.

If the macro reports ?PSTAT, ?IREC, or ?EXEC and there are no system call functions in your compiled–language sources, this means the compiler is generating the system call from high–level sources. Calls ?PSTAT, ?IREC, and ?EXEC can be made in big–PID compatible form; in fact, all these calls are big–PID compatible unless any is used in a limiting context. (See "Limiting Contexts in Calls ?PSTAT, ?IREC, and ?EXEC.")

The best course is to make sure your compiler is generating big–PID compatible code (code without limiting contexts). Read the language product Release Notice for information on big–PID compatibility, or call your Data General support organization. You may need to acquire a version of the compiler that produces big–PID compatible code; then recompile and relink the program.

After you determine that a program has no small–PID limitations, or eliminate such limitations, you're all set. You can run SPRED on the program file and label it a hybrid or anyPID program, as described in "Changing PID–Size Type."

If the Macro Reports System Call ?DADID

Call ?DADID is not restricted to small PIDs. It's acceptable in a program that has no small–PID limitations. So, if a program is compatible with big PIDs and it issues ?DADID, you can make it hybrid or anyPID at will.

However, a smallPID program issuing ?DADID can be a problem in a big–PID environment. If a C–type process runs a smallPID program, and the small–PID program issues ?DADID, then the smallPID program will receive a PID number it sees as illegal. This problem with an illegal PID number can also happen if the smallPID program issues ?DADID to any C–type process. Therefore, if a smallPID program issues ?DADID, you must make the smallPID program big–PID compatible — or, if this is impossible, make sure the smallPID program is run by and communicates with only a B–type or A–type process (not a C–type process).

Checking Assembly Language Programs

You can check assembly language program source files directly for calls ?PSTAT, ?IREC, and ?EXEC, or you can run the macro PIDCALL_CHECK on the program file. Either test will tell you whether the program issues any of these calls.

If a program doesn't issue call ?PSTAT, ?IREC, or ?EXEC, you don't need to change it. You can change the PID–size type at will (described in a later section).

If a program does issue one or more of these calls, check the next section of this chapter. If the program issues the call in a context that isn't limited, you don't need to change the program to handle big PIDs.

Limiting Contexts in Calls ?PSTAT, ?IREC, and ?EXEC

If any program uses ?PSTAT, ?IREC, or ?EXEC, the program sources must be changed if the program is to handle big PIDs.

- The program uses the ?PSTAT system call to get information on its sons. The ?PSTAT call returns status information on processes, including a list of sons. ?PSTAT can describe only sons with PIDs 1–255. ?PSTAT has no other small–PID limitation.

  If a program uses ?PSTAT and relies on the sons information returned in the ?PSTAT packet, arrange to have system call ?SONS added to the program. The program must get the sons information from the ?SONS buffer, not from the ?PSTAT packet.

- The program uses the ?IREC system call to listen for a termination or obituary message. When a process terminates, it sends a termination message to its father. If a process has connected (?CON call) to any other processes, the process also sends an obituary message to all connected processes when it terminates.

  The system places termination and obituary messages in the ?IREC packet of your program. If a program checks termination or obituary messages (perhaps it checks to see if or why the process terminated), the program must change the way it parses the message. The format of the message in the ?IREC packet has changed.

- The program uses the ?EXEC system call, function ?XFSTS. This function tells the operating system to get a process's relationship to EXEC. If a program uses call ?EXEC, function ?XFSTS, then replace ?XFSTS with an extended function, ?XFXTS.

After making the source changes needed, have the program assembled and linked using the PIDSIZE switch, or change the PID–size type with SPRED.

Details on the new and changed calls appear in the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary.* Termination and obituary messages are explained in the *AOS/VS System Concepts* manual.

Changing PID–Size Type

To change a program's PID–size type, run the SPRED program, choose option 6, "Edit PID–size type," specify the PID–size type you want, choose option 8, "Apply changes to program," and leave SPRED by typing BYE. To change a program's PID–size type, you need write access to the program file.

NOTE: Once again: Don't use SPRED to make any smallPID program into a hybrid or anyPID program until you've made sure the program doesn't have small–PID limitations. At least, run macro PIDCALL_CHECK on it to check for any potentially limiting system calls. If the macro finds any, don't change the program PID–size type until you've explored further.) For example, to change MYPROG from smallPID to anyPID, the dialog might go as in Figure 13–2.

) XEQ SPRED MYPROG.PR ↵

... (SPRED menu) ...

6. Edit PID–size type

...

Enter choice(s), separated by commas: 6 ↵ — Choose PID–size.

PID–size type [SMALLPID] (from program file)
Choices are: (SMALLPID, HYBRID, or ANYPID) new value: anypid ↵

... (SPRED redisplays menu) ...

8. Apply changes to program file

....

Enter choice(s), separated by commas: 8 ↵ — Choose Apply.

Program type initialization successful — SPRED confirms.

... (SPRED redisplays menu) ...

Enter choice(s), separated by commas: BYE ↵ — Leave SPRED, and return to the CLI.

Figure 13–2 SPRED Dialog to Change a Program's PID–Size Type

The SPRED program exclusively opens a program file; thus you can't run SPRED on a program anyone is using. To handle this, wait until no one's using the program; then run SPRED on it.

The Link program can create a program with a given PID–size type via the /PID_SIZE= value (ANYPID, HYBRID, or SMALLPID).

Hints for Using Big PIDs

- There are more PIDs above 255 than below it. Thus, it's usually beneficial to make programs anyPID, since this type will allow them to run above 255 (if PIDs are available there). If no PID is available above 255, the program will run below 255, as a B–type process.

  With most systems that run a lot of PIDs, many processes are based on one program. For example, most user processes are CLIs based on the CLI program. Also at any moment, many user processes may be SED text editors — again one program.

This general use of one Data General program can help you implement big PIDs. Most Data General programs are hybrid programs. If there are no warnings in the product Release Notice, you can free many PIDs in the low–PID range by changing one (or more) often–used Data General hybrids to anyPIDs — a simple step with SPRED. All processes from such programs will then automatically run above PID 255 — if PIDs above 255 are available — freeing PIDs below 255 for hybrid (and smallPID) programs.

For example, with 50 SED users, making SED an anyPID program will run all SEDs above 255, freeing 50 PIDs in the low range.

- Data General wants to ease the transition to big PIDs. Wherever practical, we will ship programs as anyPID programs, which run in the high range if possible preserving numbers in the low range. With any major product, like the CEO system, you should check the PID–size type (via macro PIDSIZE or the Release Notice) before planning your big–PID system. You may find, as in the example below, that Data General has anticipated your needs and shipped the program as an anyPID program.

- Generally, if you want to run many PIDs on your system, we recommend that you eliminate all small–PID restrictions from your programs (if possible). Eliminating small–PID restrictions will free you from concern about interprocess communication problems and allow you to concentrate on other demands of your application.

  By default, the Link program creates smallPID program files, although code within the program may be big–PID compatible. In a big–PID compatible system, make your site's programs anyPID or hybrid via the /PID_SIZE=ANYPID or /PID_SIZE=HYBRID switch.

- If you can't adapt all your programs for big PIDs (because a program has small–PID limitations and its source code is lost, or for any reason), then you must find a way to execute the program below PID 255, from a hybrid program. This is one reason why the CLI is shipped as a hybrid program.

- You can change most commonly used Data General hybrid programs — including text editors, but not the master CLI or PMGR — to anyPID programs. Before doing this with any Data General product, check the product Release Notice for warnings of any limitations.

- You can check any program's PID–size type with macro PIDSIZE.CLI, and check a process's PID–size type with PED and the /PIDSIZE switch, described earlier.

- If, using big PIDs, a PID–size oriented error occurs, check the appropriate message found in the *AOS/VS and AOS/VS II Error and Status Messages* manual.

Example of a Big–PID System

In this example, say you want to support 120 users, all of whom will (at one time or another) use the CEO system. Seventy users will use only the CEO system; at an average of two and one half processes per user, they will need a maximum of 175 processes. Fifty users will need the CLI as well as the CEO system; at three and one half processes per user, they also will need a maximum of 175 processes. This represents a total of 350 processes for users, so you definitely need big PIDs.

The next step is to clarify your users' needs for processes, as follows:

General User	CEO–Only User
Needs the CLI and a DGC utility program like the SED text editor or CEO Control Program and CEO Word Processor. Total averages three and one half processes.	Needs CEO Control Program and CEO Word Processor or other CEO program; occasionally needs CEO Spelling. Averages two and one half processes.
User profile has CLI as the initial program. At least three sons are allowed.	User profile has CLI as initial program, but the initial IPC file is CEO.STARTUP.CLI, which chains to the CEO system, eliminating the CLI. At least two sons are allowed.

Next, you need to decide where processes will run. Assume 30 processes for the operating system, the CEO system, and the network. This leaves 225 PIDs available below 256 for users. You need a total of 350 processes for users. Thus, you need to have a minimum of 125 processes above 255.

There are several ways to have 125 processes run above 255. One way would be to run an anyPID CLI for users (that's 50 PIDs above 255) and anyPID text editors and Sort/Merge (say another 50 PIDs above 255). This is a total of 100 PIDs above 255 — this might work, depending on what users want to do at any given time. The goal is to prevent anyone from receiving a *Too many processes* error message. Unfortunately, 100 PIDS above 255 won't always be enough.

Perhaps the CEO system will offer an alternative. Most of the processes on the system will be CEO Control Program and CEO Word Processor processes, which are any–PID, and run above 255 freeing 120 PIDs (50 CLI–CEO users and 70 CEO–only users) below 255.

The CEO Control and Word Processor programs have many interrelationships and dependencies; you can't even consider changing their PID–size type. However, you can check the PID–size type by reading the product Release Notice or running the macro PIDSIZE on the program. You find that the CEO Control Program is anyPID and the CEO Word Processor is anyPID. Having the word processor an anyPID program practically solves the problem; making Data General–supplied text editors (like SED) into anyPID programs will solve it.

The next step is to sketch the arrangement of processes for each class of user when the system runs. The sketch looks something like Figure 13–3.

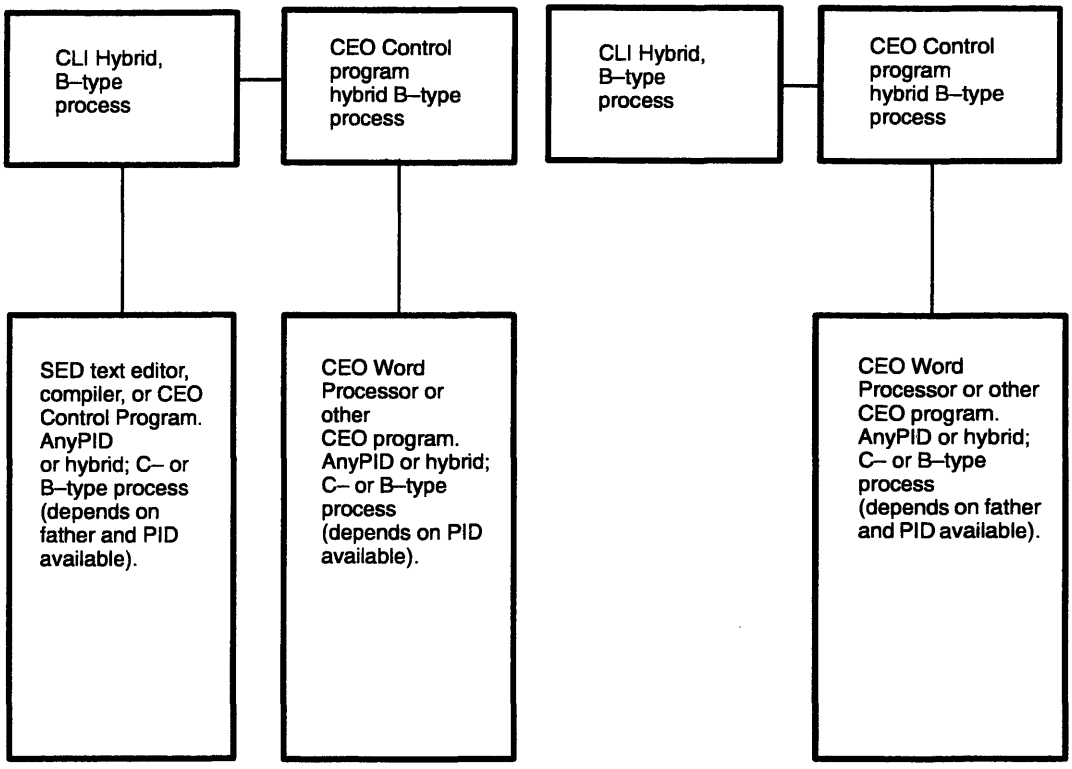


Figure 13–3 User Processes on a Big–PID System

Having planned your big–PID system, you can implement it. As Figure 13–3 above implies, many processes may be forced to run as B–type if there are no PIDs available above 255. Therefore, although you really need only 125 or so extra processes, choose a larger VSGEN maximum, like 600 processes, to permit more processes to run above 255.

Then, if necessary, you can create and/or edit users' profiles with PREDITOR. And, if you need to, you run SPRED on the SED text editor and other text editors, changing their PID–size type to anyPID.

Ultimately, when the system runs under maximum load, the PID arrangement looks like that shown in Table 13–7.

Table 13–7 Maximum Load PID Arrangement

PID	Program(s)	PID–Size Type
1	PMGR	Hybrid, B–type process.
2	CLI	Hybrid, B–type process.
3	EXEC	Hybrid, B–type process.
4–25	Network, data management and CEO control processes; CEO Control Program.	Hybrid, B–type process.
26–255	User CLI, compiler, other program development processes; CEO Control Program.	Hybrid or anyPID.
26–255	Old applications with small–PID restrictions that can't be changed (sources lost).	SmallPID, A–type process.
256–380	CEO Word Processor or other Data General text editor process.	AnyPID, C–type process.

Running an anyPID CLI for Users

If more than 200 users will have the CLI as their initial program, you will probably need an anyPID version of the CLI. (Don't make the original CLI an anyPID program, since the operating system expects the master CLI to run as PID 2.)

An anyPID CLI behaves like an anyPID program. If a PID above 255 is available, the user's CLI will run as a C–type process. If a PID above 255 isn't available, the user's CLI will run as a B–type process, like the original CLI. (All sons of B–type processes are B–type processes, so the sons may consume all PIDs below 255. You can fix this relatively easily by running VSGEN again and allowing more PIDs.)

To create and use an anyPID CLI, copy the original CLI and make the copy an anyPID program. Then run PREDITOR on selected users' profiles and insert this CLI's name as the initial program. The dialog for this, using the 16–bit CLI, would be as follows:

) SUPERUSER ON ↵	Turn Superuser on.
Su DIR : ↵	
Su) COPY/V CLI_ANY.PR CLI16.PR ↵	Copy CLI program file.
CLI16.PR	
Su) COPY/V CLI_ANY.OL CLI16.OL ↵	Copy CLI overlay file.
CLI16.OL	
Su) ACL/V CLI_ANY.PR [!USER],WARE +,RE ↵	
	Set ACL to allow access.
CLI_ANY.PR	
Su) XEQ SPRED CLI_ANY.PR ↵	Run SPRED, change PID–size type to anyPID, apply changes, exit.
Su) XEQ PREDITOR ↵	Run PREDITOR and specify :CLI_ANY.PR as initial program for all desired users.
...	
Su)	

There's one minor disadvantage to running CLIs of different PID–size types: one CLI will have a different name, thus it will not automatically be patched on updates or replaced when you install a new revision. After you do install a new revision of the operating system, delete the CLI not named CLI.PR; then copy CLI.PR; and run SPRED again as shown above.

Big–PID Summary

The operating system supports as many as 255 processes by default. You can create a system for more processes using the following steps:

1. At VSGEN, specify the desired number of processes by answering the question *Maximum number of processes*. Then build, patch, test, optionally install, and run the system as usual.

2. For each program you want to handle any PID size, check for small–PID limitations by running macro PIDCALL_CHECK.CLI as described above. If the macro finds call ?PSTAT, ?IREC, or ?EXEC, explore the code further; change and recompile if needed. If the macro doesn't find any of these calls, the program probably has no small–PID limitations.

3. Using SPRED, change the PID–size type of each program checked or changed in the previous step to hybrid or anyPID. Then apply changes to the program file and leave SPRED.

4. Don't change the PID–size type of any program supplied by Data General until until you've checked for warnings — and found none — in the product Release Notice.

Most programs shipped with AOS/VS and AOS/VS II are hybrid programs. They can execute or communicate with programs of any PID–size type, but must run at a PID between 1 and 255.

Multiple Processor Computers

Some ECLIPSE MV/Family computers, like the MV/20000 Model 2, have more than one main CPU. (In such systems, the CPUs are called job processors; the main job processor is called the mother processor; and processors other than the mother are called child processors.)

The operating system starts running on the mother processor. To enable the operating system to recognize and use the child processor(s), someone must issue the CLI command in the format

JPINITIALIZE n (n is the number of the child; e.g., 1)

This command puts the child's processing power under operating system control.

To issue JPINITIALIZE commands, a process needs the System Manager privilege. Usually, JPINITIALIZE commands are issued by the master CLI, which has all privileges, in the UP.CLI macro.

You can release a child processor via the command JPRELEASE, which again requires the System Manager privilege. Shutting down the operating system releases all job processors, which means the JPRELEASE command isn't needed for shutdown. However, you might use JPRELEASE if you wanted to remove a child processor from the system and continue running with the mother processor.

The default mother and child processors are defined in hardware, but you can change the definitions via the SCP CLI command EXAMINE. With the I (INITIAL_JP) argument, the EXAMINE command lets you view or change the default initial job processor. After you have changed the default initial processor (mother processor), reboot the system to run it on the new mother processor.

For example, the following dialog redefines the mother processor and starts bringing up the operating system on the new mother processor.

SCP-CLI / Jp0> EXAMINE INIT ↵	Type EXAMINE INIT and press NEW LINE to examine the initial job processor.
INITIAL_JP = [Jp0] 0=Jp0, 1=Jp1 > 1 ↵	The display shows the default initial job processor to be job processor 0. Type 1 and press NEW LINE to make job processor 1 the new mother.
SCP-CLI / Jp0> BOOT 24 ↵	Reboot to start up the operating system in job processor 1.

Classes and Logical Processors

Classes and logical processors allow you to give sets of processes more (or less) CPU time. They may be helpful in situations where processes need more CPU (job processor) time. They won't help in situations where processes are competing for memory (memory contention) or competing for access to disk.

Assuming processes need more CPU time, class scheduling can help in situations that require nonstandard scheduling. Such situations are those where you need to

- Give one compute–bound process priority over another compute–bound process without starving the lower priority process. (Batch jobs are examples of compute–bound processes.)
- Give interactive processes higher priority than compute–bound ones without the risk that the compute–bound processes will get no time.

A class is a set of processes for which you want special scheduling treatment. Usually, this treatment involves allotting a percentage of job processor (CPU) time. Each class has one or more user and program localities (called locality pairs). A process will run in a specific class if its user and program localities match those defined for the class.

The user locality for a process is defined with PREDITOR in a user's profile. If the profile allows, a user can change user locality (CLI command LOCALITY). The program locality is defined with the SPRED editor in the program file. User and program localities each range from 0 through 15.

A logical processor is a scheduling arrangement that usually includes a set of classes. You can move a job processor from one logical processor to another giving your site an instant new scheduling environment. By default, class scheduling is disabled.

You can create and use classes and logical processors, enable class scheduling, and monitor class use via system calls. Much more conveniently though, you can use the optional Class Assignment and Scheduling Package (CLASP).

Classes and Logical Processor Example

An example of an application of class scheduling using two logical processors — on a computer with two job processors — appears in Figure 13–4.

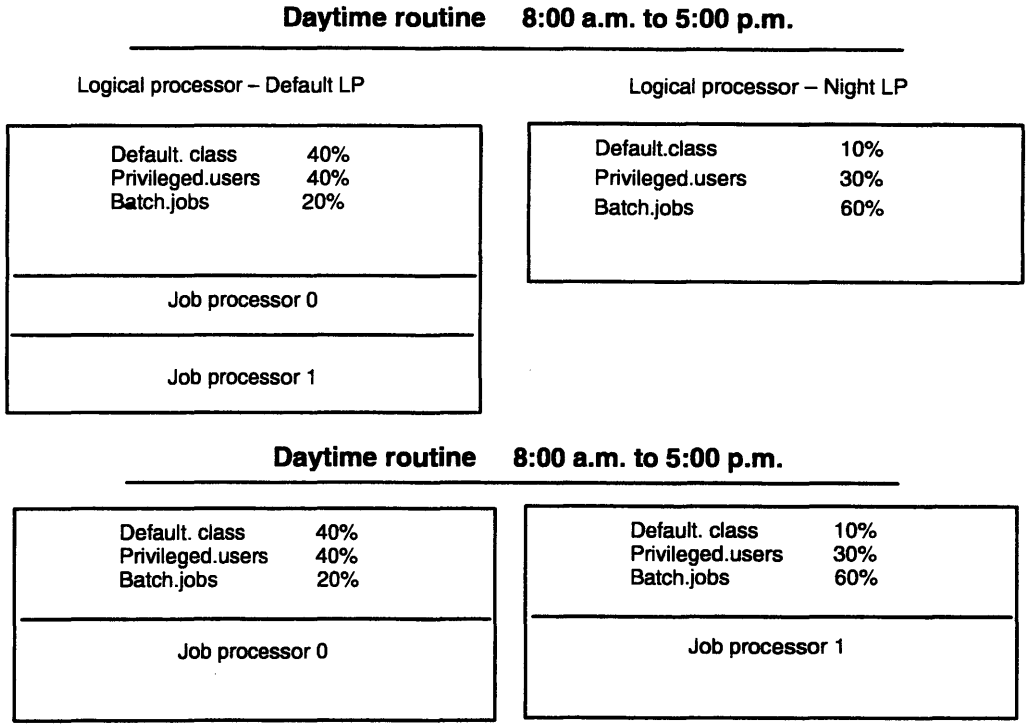


Figure 13–4 Class and Logical Processor Example

Figure 13–4 shows a logical processor that favors the default and privileged classes (interactive users) during the working day. At night, one of the job processors is moved to another logical processor giving computing time to the processes in classes on both logical processors computing time.

Disk Space and Performance

The operating system must make at least one disk access whenever any process wants to read, write, create, or delete a file. In an active system, it may make hundreds of accesses each minute.

When your system is new and its LDUs hold relatively little material, file access time is very short. As files accumulate, the operating system and the disk hardware may require more time to access files.

There are several things you can do to streamline disk access and keep performance near its optimum. There are three factors involved.

- Overall free space
- File fragmentation
- Bitmap and overlay areas

Overall Free Space

The amount of free disk space usually affects performance more than any other factor. When more than 70% of an LDU's blocks are occupied, the read/write heads must spend significantly more time moving over the disk to find free space whenever a process creates a file. Above 70% of capacity, the access time needed rises steeply. Much above 70%, everyone on the system may note slower response time. If more than 95% of an LDU is used, you should take immediate action to free some space.

The most convenient way to handle the space factor is by limiting each user's space in his or her PREDITOR profile. Chapter 2 explains several ways to approach limiting a user's space allocated. If disk space is tight, and you don't want to acquire more disk units just now, you can ask users to delete all their old .ST, .PR, and .OB files. Temporary files and break files (suffixes .TM, .TMP, and .BRK) can also be deleted; so can backup files (.BU suffix) after the file system is archived (backed up). Having users delete the files they don't need can open up a considerable amount of space.

There may also be obsolete files in :UTIL, and obsolete user directories in :UDD. You can simply delete the former via the CLI. You should dump the latter for the record; then use the PREDITOR DELETE command to delete both the obsolete profile and user directory.

Use the CLI command SPACE : (or other control–point directory name) to check number of disk blocks used and remaining. For example, on a 190–megabyte (370889–block) LDU:

```
) SPACE : 
```

MAX 370889, CURR 259662, REM 111277 — About 70% of the blocks are used.

... (time passes) ...

```
) SPACE : 
```

MAX 370889, CURR 315260, REM 55629 — About 85% of the blocks are used.

(Ask users to clean up their files and directories, deleting or backing up files to tape files where appropriate.)

```
) SPACE : 
```

MAX 370889, CURR 240430, REM 130459 — Better; under 70% of the blocks are used.

File Fragmentation

Each operating system file has one or more data elements. An element is, by default, four disk blocks, but anyone can specify many more blocks — creating a file with many contiguous blocks — with the CLI CREATE command and /ELEMENTSIZE switch. (Under AOS/VS II, you can run Disk Jockey and choose different default file element sizes.)

After an LDU begins to fill up, files become fragmented: the operating system must search farther on the disk for space for new file elements. For example, one element could be near an outer cylinder, and the next available space near an inner cylinder. To write the next element, the system must move the read/write head all the way across the disk.

File fragmentation often occurs when an LDU is nearly full. Simply deleting files (as above) may or may not eliminate it. If cleaning up the LDU doesn't help, and the LDU's disk drives appear to be doing a lot of seeks, you can suspect fragmentation. You can also use the DISCO program described in Chapter 11.

To minimize or eliminate fragmentation, clean up the LDU as above. Then dump all files from the LDU and reformat the disk(s) using Disk Jockey as described in the manual *Installing, Starting, and Stopping AOS/VS II*. Make sure the bitmap and overlay area are near the middle of the first disk. Then reload all the files and see if performance picks up. It usually will.

You can always eliminate fragmentation of a specific file by creating it with a large element size, but this is impractical and inefficient for most files. It is useful — and recommended — for database files that will be used by data management systems like INFOS II.

Bitmap and Overlay Areas

The operating system must write to the bitmap every time it allots a new disk block to a file. It must access the overlay area every time it needs a system page that isn't already in memory. Thus the addresses of these tables can affect performance.

During the formatting that creates an LDU by default, the Disk Formatter and Disk Jockey put the bitmap 1/3 of the distance across the first disk in the LDU. For a system disk, the overlay area follows the bitmap. Disk Jockey defaults are good general–purpose addresses.

If you use a multiple–disk LDU for a very large file (like a database file that won't fit on one disk), the best place for the bitmap is at the beginning of the LDU. This allows for the largest contiguous amount of file space.

You can check the bitmap and overlay area addresses with Disk Jockey. If you decide to move the bitmap and overlay area (if any) for better performance, and you've been using the LDU for a while, the area you want is probably already allocated to files. The best course is to dump all files from the LDU, run the Disk Formatter or Disk Jockey, put the bitmap (and overlay area) where you want, and reload the files. This procedure will also reduce fragmentation on the LDU.

Data Caching on AOS/VS II LDUs

AOS/VS II can reserve a portion of main memory to store disk information, based on the premise that the information may be used again soon. AOS/VS does not have this capability. Access time to information held in memory is much faster than the amount of time it takes to access the same information from disk. The portion of main memory saved for this purpose is called a data cache or cache. Because of faster access times, a data cache has the potential to increase system performance.

Once your AOS/VS II system is up and running, the fact that the data cache exists will be invisible to you. You do not have to change any of your .PR files to use data caching. Any application that you are running without data caching can be run with data caching.

There can only be one data cache per AOS/VS II system, but participation in the data cache is on an LDU basis; that is, not all LDUs on a system must use data caching. You turn the cache on or off by answering the series of data caching questions that are asked at VSGEN time. You can choose if an LDU is to use data caching by specifying the /CACHE switch on the CLI INITIALIZE command when you initialize the LDU. The form of the command is

```
INITIALIZE/CACHE
```

(Type HELP/V INITIALIZE and press NEW LINE for an on–line Help file that explains the /CACHE switch.)

The main measure of a data cache's efficiency is the cache hit ratio; that is, how often an LDU accesses the data cache. To help you fine tune your system, Data General offers the AOS/VS and AOS/VS II Performance Monitor (which is sold as a separate product), that maintains metering information concerning the data cache. The AOS/VS and AOS/VS II Performance Monitor can show you the hit ratio for individual LDUs.

Even if you do not have the Performance Monitor software, you can still experiment with data caching and, by timing your applications, determine an efficient size for the data cache. Or you may decide not to using caching because it offers no performance improvement for your system.

Data General database management products like DG/SQL and INFOS II use their own data caching mechanisms, but they can still benefit from an AOS/VS II system cache.

You may want to consider using a system cache in addition to the caching mechanisms your database products provide because large cache entry sizes offer better look–ahead benefit than database products with page or (in the case of DBMS) block granularity. The system cache size is configurable while the DBMS cache size is not.

And, the system cache can be shared by all processes, while the database caches are per process or per file.

Data Cache Format and Organization

Enabling data caching is done through a series of VSGEN questions. Activating data caching occurs through CLI commands. However, it's necessary to have an understanding of the organization of the cache to use it properly.

The data cache is allocated as one logically contiguous piece of memory at system boot time. The cache is divided into equal pieces called *entries*. You set the cache size and entry size via VSGEN.

There is one cache that is global to all LDUs that are initialized for data caching.

User Specifiable Parameters via VSGEN

You can specify LDU data caching at VSGEN time and also at the *Override default specifications* question when booting the system disk. User specifiable parameters allow you to enable caching, set the cache size in megabytes, set the cache entry size in blocks, and enable caching for the root LDU.

The VSGEN process is not within the scope of this manual. The specific questions are described fully as part of the discussion of VSGEN in the *Installing, Starting, and Stopping AOS/VS II* manual.

Data Caching Considerations

Two questions that you may want to ask yourself if you are considering using LDU data caching are

- Given my current configuration, what effect will caching have?
- Should I buy more memory to hold the cache?

The first question can be answered by trying data caching and observing the effect, either directly or with the AOS/VS Performance Monitor software. Since data caching has no ill side–effects other than the possibility of reduced performance, you can feel safe in giving caching a try, and observe what happens (to system performance). You can also try simulated data caching. If you decide against using data caching, you can disable caching (as described in the section on VSGEN in the manual *Installing, Starting, and Stopping AOS/VS II*).

The second question is more difficult to answer since allocating a cache in effect takes away physical memory that could be used for other purposes, thereby changing the operational properties of your machine; for example, causing more paging and swapping.

As an example, take a machine that has 10 Mbytes of memory. I/O is a major bottleneck on this machine. The machine is not yet paging and swapping, but it is close to it; there are seldom free pages in the free chain. You decide to try caching. Knowing that you don't have too much memory to spare, you try running with a 1 Mbyte cache at first. The machine is running several applications at once and although each application has a reasonable locality of reference, the system could really use about 1 Mbyte of cache per application. Disappointed with the performance of the 1 Mbyte cache, you try a larger cache. Unfortunately, this causes paging and swapping. At this point, you may find yourself considering purchasing more memory to run the data cache.

Data Cache Simulation

For just such cases, Data General created cache simulation. The cache simulation is set up and works just as the data cache normally would. The only difference is that no memory space is allocated for the cache entries; no caching is actually done. All the statistics that would have been generated by running a real data cache of the user–specified size are still calculated and most of the CPU overhead for manipulating the data cache occurs (except for the actual moves of data in and out of the cache). The system manager can then examine the hit rate using the AOS/VS Performance Monitor, and make an approximation of how the system would operate if more memory were purchased specifically for the data cache.

If you choose to run the cache simulation, there are three points to keep in mind:

- The cache simulation will only be an approximation. There are several reasons for this, perhaps the most important is that the cache simulation does not really move any data in or out of the cache. It is conceivable, though unlikely, that the cache simulation could overstate the usefulness of a data cache.
- The cache simulation can only provide information on cache hits; it cannot tell you how much faster your system will run. You must deduce the actual benefit to be derived from a data cache by an analysis of the data provided by the AOS/VS and AOS/VS II Performance Monitor, or by runtime experience with actual (not simulated) caching.
- Cache simulation and actual data caching are mutually exclusive; they cannot both be run at the same time.

Using the HISTO and HISTOREPORT Utilities

The operating system, via the ?WHIST system call, allows you to perform a histogram against either a single process or against all processes. The operating system will return raw histogram data in a pre–determined format.

Histograms record the number of clock periods or *ticks* spent in specific logical address ranges or *buckets* over the histogram duration. At the end of each tick of the clock, the operating system evaluates the value of the program counter and determines which bucket counter is to be incremented. While histograms do not provide exact information and though a statistical deviation must be anticipated, histograms do provide an indication of how much process run time is being consumed within the specific bucket ranges.

The system calls ?WHIST and ?KHIST permit the user to histogram a user–specified logical address range. With this information, the user has the means to identify and rectify inefficient sections of his program's code. Through repetitive histograms over increasingly smaller intervals, measurements of the smallest intervals may be made.

The histogram data may indicate that an excess amount of time is being spent in a particular routine. This would suggest that an optimization of this routine would significantly improve program performance.

The utilities HISTO and HISTOREPORT work together to furnish the histogram information to the user. HISTO executes the ?WHIST system call using specifications you provide, and generates a file containing raw data. Then HISTOREPORT analyzes the file that the HISTO utility generates, and produces a report file containing the interpreted histogram information, that the user can read.

HISTO/HISTOREPORT support includes

- 16–bit and 32–bit target processes.
- Full ring support. Addresses in rings 0 through 7 may be monitored, and symbol table files may be provided for any ring.
- Clock tick totals for various segments of the system, the target process, and other processes.
- A report detailing the limits of each bucket, the number of ticks recorded in that bucket, and the number of ticks expressed as a percentage of the total ticks recorded within the whole histogram range.

HISTO/HISTOREPORT optionally provides symbolic representations for the starting logical addresses of each bucket.

The Data Collection Utility (HISTO)

HISTO executes the ?WHIST call on the target process and, writes the raw histogram data to a file in a form intelligible to the report program (HISTOREPORT.PR).

You specify the parameters defining the target process(es), how long you want HISTO to run, the logical starting address of the area to be analyzed, the respective logical ending address, the size of each bucket within this range, and the name of the file to receive the data.

The HISTO needs to become resident; this is the only restricted privilege required.

HISTO's Parameters

Before HISTO can be invoked, the target process must already be initialized, because HISTO requires the target process's pid number as an initial argument.

You can invoke the HISTO utility with the following command line:

```
XEQ HISTO pid# run saddr eaddr bucket datafile
```

Parameters to HISTO are specified as arguments on the command line. There are six required arguments:

pid#	The pid of the target process, or zero if the target is "all processes".
run	Maximum histogram duration, in seconds. HISTO will not terminate until the specified number of seconds has elapsed, even if the target process terminates.
saddr	The starting address of the area of the target process. This value must be specified in octal and must include a ring field; a ring field of zero indicates a ring zero address. This value corresponds to offset ?HWST in the ?WHIST packet.
eaddr	The ending address of the target process. This value must be specified in octal and must include a ring field; a ring field of zero indicates a ring zero address. This value corresponds to offset ?HWEND in the ?WHIST packet.
bucket	The size of each interval in the histogram area. This value must be specified in octal. This value corresponds to offset ?HWWDS in the ?WHIST packet.
datafile	Pathname of the file to receive the histogram data. This file must not already exist.

The Data Analysis and Report Utility (HISTOREPORT)

This program interprets the raw histogram data logged by HISTO, and generates a report file in user–readable format. This report file consists of a report header followed by a histogram chart.

HISTOREPORT may be invoked by the following command line:

XEQ HISTOREPORT/I=pathname/O=pathname *[switches]*

Parameters to HISTOREPORT are specified as switches on the pathname argument in the command line. Unique abbreviations and both upper– and lowercase are acceptable. These switches enable you to control the input/output to and from the utility.

Switch	What It Does
/I=pathname	Read the data from file PATHNAME. This file must be a data file created by HISTO.PR. This switch is required.
/O=pathname	Write the generated report to file PATHNAME. This file must not already exist. This switch is required.
/ST	Access the default symbol table file when determining symbolic representations of ring 7 logical addresses. The default for single–process targets is root pathname.ST; there is no default for all–process targets.
/ST=pathname	Access the specified file PATHNAME when determining symbolic representations of ring 7 logical addresses.
/R7ST	Same as /ST. Provided for conformity to the rest of the symbol table switches.
/RxST=pathname	Access the specified file pathname when determining symbolic representations of ring *x* logical. Each LDU can also be thought of as being divided into a series of equal size contiguous pieces of size m beginning at block 0 of the LDU. These pieces are called logical disk frames. Cache entries hold logical disk frames. *x* may be any number between 0 and 7 inclusive.

HISTOREPORT Output Format

HISTOREPORT produces a report file consisting of a header and a detailed report.

The header of the HISTOREPORT file will supply some or all of the following information:

- Target Process name
- Target Program pathname
- Ring 7 Symbol filename (if specified)
- Ring 6 Symbol filename (if specified)
- Ring 5 Symbol filename (if specified)
- Ring 4 Symbol filename (if specified)
- Ring 3 Symbol filename (if specified)
- Ring 2 Symbol filename (if specified)
- Ring 1 Symbol filename (if specified)
- Ring 0 Symbol filename (if specified)
- Date/Time of Histogram initialization
- System statistics
- Process runtime statistics
- Starting address of histogramed area
- Ending address of histogramed area
- Interval (bucket) size
- Number of intervals
- Total clock ticks during histogram
- Ticks in the target process within histogram area
- Ticks in the target process outside histogram area
- Ticks in other processes
- Ticks in the system (excepting idle)
- Ticks in the system idle loop

Information Provided by HISTOREPORT

A detailed report providing statistics for each nonzero bucket within the histogramed area is included in the report file. This report contains the following information:

- Ring containing logical address of bucket start
- Logical address of bucket start
- Count of ticks within the bucket
- Cumulative tick count
- Percentage of ticks in bucket as a function of total
- Ticks within histogramed area
- Cumulative percentage
- Symbolic representation of bucket starting address (optional)

General Notes

The limit on the number of buckets to 20000 decimal is a restriction imposed by these utilities, not by the system. AOS/VS requires that the utility be resident for the duration of the histogram, and the pages containing the buckets counters are implicitly wired into memory by the system during the processing of ?WHIST. As a result, the bucket counter pages will neither swap out nor page out, increasing system memory contention. Additional buckets were determined to unnecessarily penalize system performance.

There is no limit on the size of the buckets. Buckets may cross ring boundaries. Also, the buckets may contain addresses invalid to the target process.

Histograms may be run from the very initialization of the target process. This is accomplished by issuing a ?PROC with the ?PFBS flag set; this will cause the target to be blocked immediately after it is created. A histogram may then be activated and the target unblocked. The immediate blocking of the target may also be accomplished through the CLI utility by specifying the switch /BSON on the PROCESS command.

HISTOREPORT Examples

Examples of each of the two output modules, Header and Detailed Report, are provided below. These two modules are the complete product of a single report file.

Header Example

Figure 13–5 shows a HISTOREPORT Header example, edited to fit the format constraints of this chapter. All changes were cosmetic; no functionality has been removed.

```
Histogram Report Utility -- Revision n.n.n.n 12-May-93 12:50:18


Process name:      BRINT:020
Program pathname: :UDD:BRINT:HISTO:TEST.PR ( 16-Bit Process )

Ring 7 Symbol File:      TEST.ST
Ring 6 Symbol File:      - not specified -
Ring 5 Symbol File:      - not specified -
Ring 4 Symbol File:      - not specified -
Ring 3 Symbol File:      - not specified -
Ring 2 Symbol File:      - not specified -
Ring 1 Symbol File:      - not specified -
Ring 0 Symbol File:      - not specified -

Date/Time of Histogram:        12-May-93 12:49:40 PM

System Statistics:

     System Revision           03.00.00.00
     System Identifier         AOS/VS REV 3.00.00.00
     Memory Configuration      4.00 megabytes

Process Runtime Statistics over Histogram:

                     Start      End             Total
Elapsed Time          24        54              30 seconds
CPU Time              20451     48473           28022 milliseconds
I/O Usage             0         0               0 blocks
Page-Seconds          143       339             196

Histogram Header/Packet Information:

Starting address of monitored area:  16000000000 (octal)
Ending address of monitored area:    16000017777 (octal)
Interval size:                       00000000001 (octal) words
Number of Intervals:                        8191 (decimal)
```

Figure 13–5 HISTOREPORT Header Example (continued)

```
Total clock ticks during histogram:          300
Ticks in target inside range:                278
Ticks in target outside range:                 1
Ticks in other processes:                     21
Ticks in system (excepting idle):             10
Ticks in system idle loop:                     1
```

Figure 13–5 HISTOREPORT Header Example (concluded)

Detailed Report Example

Figure 13–6 shows a HISTOREPORT Detailed Report example. Like the previous example, the following example has been edited to fit the format of this chapter. All changes were cosmetic; no functionality has been `removed`.

```
      Bucket       #         Cum       Rel       Cum    Symbolic
Ring  Address      Ticks     Ticks     %         %      Representation
-------------------------------------------------------------------

     -----
(7)   16000000451 175       175       62.94     62.94    LOOP
(7)   16000000452  57       232       20.52     83.46    LOOP+1
(7)   16000000453  46       278       16.54     100.00   LOOP+2
     -----
```

Figure 13–6 HISTOREPORT Detailed Report Example

Using the LOGCALLS Utility

Using the ?LOGC system call, a user can receive a detailed log of all system calls issued either by or on behalf of the calling process. All system calls go through the system, including task calls, resource calls, etc. The log produced by ?LOGC may also include system calls made by the system on behalf of a user system request. The log produced by this call is written to the disk in a predetermined format.

The LOGCALLS utility reads this logfile generates a report file containing the interpreted ?LOGC data. Through user–defined parameters, the report can be tailored to the user's needs.

LOGCALLS includes the following:

- Support for 16–bit and 32–bit processes.
- For each system call issued, a summary report indicating the frequency of call issuance from each of rings 3, 4, 5, 6, and 7. Provides call totals.
- Mnemonic representations of all system calls.
- Full user–ring support. Fully supports user rings 4, 5, 6, and 7.

LOGCALLS optionally provides the following:

- A report detailing the calls issued by all tasks.
- Reports detailing the calls issued by each task.
- Symbolic representations for all logical addresses from which the system calls were issued.
- Full ring 3 (AGENT) support. Reports system calls issued by the AGENT on behalf of the user.

The generated report file consists of a report header followed by a chart of all system calls in chronological order. For each system call, this chart supplies the system call mnemonic and octal representation, the accumulator values (AC0,AC1 & AC2) and program counter at system call issuance, and the task ID and the TCB number (indicating the order in which the tasks were initialized). Provides the symbolic representations of all program counters at user request.

The LOGCALLS Command Line

You invoke the LOGCALLS utility by using the following command line:

XEQ LOGCALLS/I=pathname/O=pathname *[switches]*

You specify parameters to LOGCALLS as switches on the first argument in the command line. Unique abbreviations and both upper– and lowercase are acceptable. Switches are grouped into two categories: those that control the I/O to and from the utility, and those that internally control the generation of reports. All switches are optional except for the /I= and the /O= switches.

The following switches control the input and output of the utility.

/I=pathname	Read the data from file pathname. This file must be a data file created by the ?LOGC supervisor call. This switch is required.
/O=pathname	Write the generated report to file pathname. This file must not already exist. This switch is required.
/ST	Access the default symbol table file when determining symbolic representations of ring 7 logical addresses. The default filename is <root pathname>.ST.
/ST=pathname	Access the specified file pathname when determining symbolic representations of ring 7 logical addresses.
/R7ST	Same as /ST. Provided for conformity to the rest of the symbol table switches.
/R7ST=pathname	Same as /ST=pathname. Provided for conformity to the rest of the symbol table switches.
/R6ST=pathname	Access the specified file pathname when determining symbolic representations of ring 6 logical addresses. There is no default pathname.
/R5ST=pathname	Access the specified file pathname when determining symbolic representations of ring 5 logical addresses. There is no default pathname.
/R4ST=pathname	Access the specified file pathname when determining symbolic representations of ring 4 logical addresses. There is no default pathname.
/R3ST	Access the default AGENT symbol table file when determining symbolic representations of AGENT (ring 3) logical addresses. The default is :AGENT.ST.
/R3ST=pathname	Access the specified file pathname when determining symbolic representations of AGENT (ring 3) logical addresses.

The following switches internally control the generation of reports.

/ALL	Produce a comprehensive report of all calls made by all tasks.
/EACH	Produce task–specific reports for each task. This would be equivalent to specifying the "/TASK=" switch for every task.

/LAST=n — For all reports generated, reference only the last *n* system call entries in the data file. If *n* is zero, no reports will be produced; if *n* is greater than the total number of entries, all system calls will be reported. This switch makes it easy to examine the last few system calls in a very large data file. Note that the Summary Report is unaffected by this switch.

/NOSYS — Do not log system calls made by the system on behalf of the user. When processing a system call requested by the user, the system will frequently issue additional system calls on behalf of the user; this is a normal and necessary feature of the processing of some system calls. By default, these calls issued by the system will be logged; in the report they will be prefixed by a "o" instead of the usual "?".

/TASK=n — Produce a report of those system calls executed by or on behalf of the task with the ID "n". This TID value for all tasks is available to the user in the all–task report.

LOGCALLS Output Format

LOGCALLS produces a report file consisting of a header, a system call summary chart, and a report or reports for each of the switches (/ALL, /EACH, and /TASK=) selected. The content of the three sections of the LOGCALLS report is described below.

The Header

The header contains the following information:

- Target Process name
- Target Program pathname
- Ring 7 Symbol filename (if specified)
- Ring 6 Symbol filename (if specified)
- Ring 5 Symbol filename (if specified)
- Ring 4 Symbol filename (if specified)
- Ring 3 Symbol filename (if specified)
- Date/Time of ?LOGC initialization

The Summary Report

The summary report of system calls is analyzed by system call and by protection ring, and contains the following information for each call issued:

- System call mnemonic
- System call octal value
- Count of system call issued in ring 3
- Count of system call issued in ring 4
- Count of system call issued in ring 5
- Count of system call issued in ring 6
- Count of system call issued in ring 7
- Total count of system call issued from all rings

The summary report also provides the following information for all system calls:

- Total system calls issued from ring 3
- Total system calls issued from ring 4
- Total system calls issued from ring 5
- Total system calls issued from ring 6
- Total system calls issued from ring 7
- Total system calls issued from all rings

The Detailed Report(s)

You can request detailed information concerning each system call issued by specifying the optional switches /ALL, /EACH, or /TASK=. In the report, the system calls are listed in chronological order.

For each issuance, the following information is provided:

- System call mnemonic
- System call octal value
- Accumulator 0 value at call issuance
- Accumulator 1 value at call issuance
- Accumulator 2 value at call issuance
- Program Counter at call issuance
- Task ID of task issuing call
- TCB number (indicating order of task initialization)
- Symbolic representation of the Program Counter (optional)

General Notes

The format of the report file is designed to fit within 80 columns if no symbolic representations are requested. If symbolics are requested and if the symbol table file(s) can be accessed, the report width will exceed 80 columns.

The AGENT and user program symbol table files are accessed independently; the user specifications may legitimately call for access to one symbol table file and not others. Alternatively, symbol table files for all the rings 3, 4, 5, 6 and 7 may be opened. Access to a particular symbol table file is not required.

If a specified symbol table file is never accessed (if no system calls were issued from that ring), then the specification will effectively be a no–op.

The ?LOGC system call provides the option of setting "forced output" on the logging of system call data. The LOGCALLS utility uses this functionality to ensure that system call data will be written out to the data file before the system call is processed. Consequently, should the user process abruptly terminate, whether through a user ring trap, a system ring 3 AGENT trap, or an AOS/VS fault or system panic, the data file will correctly reflect the system call sequence up to and including the last system call issued.

LOGCALLS Examples

Examples of each of the three output modules (Header, Summary Report, and Detailed Report) are provided in the sections below. These three modules are the complete product of a single report file. The examples were edited to fit the format of this manual. All edits were cosmetic; no functionality is missing.

Sample Header

```
System Call Logger -- Revision nn.nn.nn   12-May-93   12:11:05 PM


Process name:          BRINT:053
Program pathname:      :UDD:BRINT:SHP_VS:SHP_VS.PR (32-Bit Process)

Ring 7 Symbol File:  SHP_VS.ST
Ring 6 Symbol File:  SHP_VS6.ST
Ring 5 Symbol File:  SHP_VS5.ST
Ring 4 Symbol File:  SHP_VS4.ST
Ring 3 Symbol File:  :AGENT.ST

Date/Time of ?LOGCALL initialization:  12-May-93      12:06:17 PM
```

Sample Summary Report

A report detailing the calls issued by a particular task.

Mnemonic	Call #	Ring3	Ring 4	Ring 5	Ring 6	Ring 7	Total
?MEM	3	0	2	2	2	2	8
?MEMI	14	0	1	1	1	1	4
?ILKUP	27	2	0	0	0	0	2
?SSHPT	44	0	1	1	1	1	4
?RPAGE	45	0	1	0	0	0	1
?GOPEN	56	4	0	0	0	0	4
?GCLOSE	57	2	0	0	0	0	2
?SPAGE	60	0	1	0	2	0	3
?SOPEN	63	0	0	0	0	5	5
?GSHPT	73	0	1	1	1	1	4
?FSTAT	77	0	0	0	0	2	2
?PNAME	116	0	0	0	0	1	1
?IS.R	142	6	0	0	0	0	6
?RINGLD	264	0	0	0	0	3	3
?OPEN	300	0	0	0	0	2	2
?READ	302	0	0	0	0	1	1
?WRITE	303	0	0	0	0	1	1
?TASK	500	0	0	0	0	2	2
?MYTID	521	0	0	0	0	5	5
Ring Totals:		14	7	5	7	27	60

Sample Detailed Report

The following example has been edited to fit in this document. Specifically, the columns for "Ac1" and "Ac2" have been removed. These columns are identical to the "Ac0" column.

```
Mnemonic Call  Ac0           PC          TID TCB Ring   PC Symbol
-------------------------------------------------------------------
?PNAME   116   00000000000 16001766172 1     1   (7)   P.INIT+33
?FSTAT   77    34000017732 16001766211 1     1   (7)   P.INIT+52
?FSTAT   77    34000020060 16001766235 1     1   (7)   P.INIT+76
?MEM     3     00000000001 16001766276 1     1   (7)   P.INIT+137
?MEMI    14    00000000001 16001766321 1     1   (7)   P.INIT+162
?MEM     3     00000000001 16001766331 1     1   (7)   P.INIT+172
?GSHPT   73    00000000762 16001766350 1     1   (7)   P.INIT+211
?SSHPT   44    00000000007 16001766377 1     1   (7)   P.INIT+240
?RINGLD  264   34003744130 16001766565 1     1   (7)   P.INIT+426
?RINGLD  264   34003744114 16001766577 1     1   (7)   P.INIT+440
?RINGLD  264   34003744100 16001766611 1     1   (7)   P.INIT+452
?MEM     3     34003744100 14001774054 1     1   (6)   INIT+4
?MEMI    14    00000000001 14001774077 1     1   (6)   INIT+27
?MEM     3     00000000001 14001774107 1     1   (6)   INIT+37
?GSHPT   73    00000000766 14001774125 1     1   (6)   INIT+55
?SSHPT   44    00000000010 14001774160 1     1   (6)   INIT+110
?MEM     3     34003744100 12001774054 1     1   (5)   INIT+4
?MEMI    14    00000000001 12001774077 1     1   (5)   INIT+27
?MEM     3     00000000001 12001774107 1     1   (5)   INIT+37
?GSHPT   73    00000000766 12001774125 1     1   (5)   INIT+55
?SSHPT   44    00000000010 12001774160 1     1   (5)   INIT+110
?MEM     3     34003744100 10001774054 1     1   (4)   INIT+4
?MEMI    14    00000000001 10001774077 1     1   (4)   INIT+27
?MEM     3     00000000001 10001774107 1     1   (4)   INIT+37
?GSHPT   73    00000000766 10001774125 1     1   (4)   INIT+55
?SSHPT   44    00000000010 10001774160 1     1   (4)   INIT+110
?TASK    500   00000003724 16001766721 1     1   (7)   P.INIT+562
?OPEN    300   00000000005 16001770437 1     1   (7)   P.READ+4
?GOPEN   56    34003744160 06001633557 1     1   (3)   AOPEN+570
?MYTID   521   00000000000 16001767160 6     6   (7)   P.MAIN+2
```

Sample Detailed Report (continued)

Mnemonic	Call	Ac0	PC	TID	TCB	Ring	PC Symbol
?SOPEN	63	34000017732	16001767261	6	6	(7)	P.MAIN+103
?MYTID	521	00000000000	16001767160	5	5	(7)	P.MAIN+2
?SOPEN	63	34000020060	16001767261	5	5	(7)	P.MAIN+103
?GCLOSE	57	00000000003	06001635320	1	1	(3)	OPERBAD+500
?GOPEN	56	14000052226	06001635367	1	1	(3)	OPERBAD+547
?MYTID	521	00000000000	16001767160	4	4	(7)	P.MAIN+2
?SOPEN	63	34000017732	16001767261	4	4	(7)	P.MAIN+103
?MYTID	521	00000000000	16001767160	3	3	(7)	P.MAIN+2
?SOPEN	63	34000020060	16001767261	3	3	(7)	P.MAIN+103
?MYTID	521	00000000000	16001767160	2	2	(7)	P.MAIN+2
?SOPEN	63	34000017732	16001767261	2	2	(7)	P.MAIN+103
?ILKUP	27	14003472202	06001635060	1	1	(3)	OPERBAD+240
?IS.R	142	00000000001	06001635111	1	1	(3)	OPERBAD+271
?IS.R	142	00000004200	06001635233	1	1	(3)	OPERBAD+413
?TASK	500	00000000005	16001770451	1	1	(7)	P.READ+16
?OPEN	300	36000012575	16001775763	1	1	(7)	AINB+26
?GOPEN	56	34000025276	06001633557	1	1	(3)	AOPEN+570
?WRITE	303	00000000000	16001771033	0	7	(7)	P.DISP0+157
?IS.R	142	37777777777	06001600550	50	7	(3)	.PIO+627
?SPAGE	60	10001642000	10001774773	6	6	(4)	P.SPAGE+141
?GCLOSE	57	00000000003	06001635320	1	1	(3)	OPERBAD+500
?GOPEN	56	14000052210	06001635367	1	1	(3)	OPERBAD+547
?ILKUP	27	14003472202	06001635060	1	1	(3)	OPERBAD+240
?IS.R	142	00000000001	06001635111	1	1	(3)	OPERBAD+271
?IS.R	142	00000004100	06001635233	1	1	(3)	OPERBAD+413
?READ	302	36000012575	16001775767	1	1	(7)	AINB+32
?IS.R	142	00000000000	06001600550	1		(3)	PIO+627
?RPAGE	45	10000212000	10001775646	5	5	(4)	P.RPAGE+7
?SPAGE	60	14001214000	14001774773	4	4	(6)	P.SPAGE+141
?SPAGE	60	14000064000	14001774773	3	3	(6)	P.SPAGE+141

End of Chapter

Chapter 14
Maintaining System Security

Read this chapter

- When you want to maintain security with any AOS/VS or AOS/VS II system, and especially if you want to run an AOS/VS II system at the C2-level of trust (C2 is a classification used by the U.S. Government; see the "C2-Level Systems" section below);
- When you want to understand how user privileges and access control prevent system break-ins and abuse;
- When you need some guidance when running detailed system logging;
- When you want tips about running a secure system.

The goal of this manual is to assist you in managing your operating system. Its purpose is not to tell you how to manage, but rather to give you information and suggestions that can help you make good management decisions.

Major sections within this chapter are

- Security Summary
- Open or Closed Shop?
- C2-Level Systems
- How to Create a C2-Level System — A Summary
- Operating System Security Features
- User Privileges and Security
- Logon Procedures and Security
- Controlling Access with ACLs
- Auditing with the System Log
- Protecting the System Site and Backup Media
- System Architecture — Hardware Protection Features
- Security Policies
- Detecting and Responding to Breaches of Security
- Deleting a User Profile (Revoking an Account)
- Security Check List

Security Summary

The operating system has certain security features built in; others are optional — they should be used if a very secure system is desired.

In any standard system, the typical user

- must have a valid account to log on;
- cannot access another user's files (unless that user gives access explicitly by changing file ACLs or by changing his or her default ACL);
- cannot delete or modify system files;
- cannot add files to system directories;
- cannot examine data in deleted files or previously used memory pages;
- cannot edit his or her profile to become privileged.

Thus, the default privileges and access controls will protect the system from casual trespass or file violation. If more or less security is required, system flexibility permits adding or relaxing controls.

You can also have the system keep a detailed log with which you can — periodically — create reports and check user activities. For example, you can have every logon and file access attempt recorded for specified users, or for every user.

The *physical* security of the system console, central processor, and disk units is another important aspect of system security. Anyone who can touch the computer or disk units can flip a switch, bring the system down, run ESD, then restart and come up in the master CLI. The only way to protect key peripherals is to lock them up and/or keep a reliable system operator on duty at all times. In any organization you must be able to trust the people who have physical access to the computer hardware.

Open or Closed Shop?

From the standpoint of security, computer sites range from casual, relaxed *open shops* to tightly controlled *closed shops*. In the center are *medium-security shops*. In reality, of course, there are more than three variations — the definitions used here are meant as general examples.

Table 14-1 offers perspective on security for your own site. In this table, "users" represent the average user, not every user.

The actions any user can perform are governed by his or her user profile, created by the system manager with a program named PREDITOR. Privileges that can be granted in user profiles are explained later in this chapter, in Table 14-2.

Table 14-1 is just a guideline. If most of the actions described are okay, then your security needs are low: an open shop will suit you. If you want to prevent users from doing most of the actions described, then your security needs are high: you need a closed shop.

Table 14-1 User Action and Security Levels

In Your Ideal System	Open	Medium-Security	Closed
Can users access files without leaving tracks (records in a log file)?	Yes	Maybe	No
Can one account be used by the general public (is there a public account)?		Yes	No (may not be needed)
Can users learn the names of other users' files?	Yes	Maybe	No
Can remote users learn the names of other users' files?	Yes	No	No
Can users read and copy other users' files?	Yes	Maybe	No
Can users read other users' electronic mail?	Yes	No	No
Can users change or delete other users' files?	Yes	No	No
Can users gain access to tape or diskette units without operator intervention?	Yes	Maybe	No
Can users consume system resources by writing personal letters or playing games?	Yes	Maybe	No
Can users log on as other users?	Maybe	No	No
Can users give themselves additional privileges?	Maybe	No	No

The closed/medium-security/open shop issue is a management call; we can't tell you which way to go. Ultimately, you must balance the risk of losing critical information — causing delays and possibly jeopardizing deadlines, projects, corporations, or even people — against the cost and inconvenience of maintaining security. The files and information on your system may be even more important than the hardware. The general rules of risk analysis apply.

For more discussion of the issues and procedures involved in setting or changing security policy, see the sections, "Choosing Your Level of Security" and "Changing Security Levels" later in this chapter.

The Open Shop

The category of open shop includes systems like single-user and graphics workstations — any system where system users don't need privacy or fear deletion of files or vandalism.

In an open shop, the hardware and system console may be open to programmers and even nonusers. Usually, a lockable CLI is not run on the system console. Access control for the tailored system may not be enabled (enabling access control is a VSGEN option; you must enable it to have operating system control access).

Generally, if you want to run an open shop (as defined here), you don't really need to read this chapter. Possibly, though, you might want to see how much work is involved in running a secure system, or learn about common break-in tactics. You might also want some information on full detail logging (see the "Auditing with the System Log" section).

The Medium-Security Shop

The medium-security shop wants some measure of security, without the inconvenience and performance penalties of a closed shop. At least some users require privacy and protection against file deletion and vandalism. Physical access to the computer and system console is more restricted than in an open shop.

A medium-security shop may or may not have a system operator. If not, a lockable CLI will be run on the system console. Several people, but not all, may have Superuser privilege, and/or know the locked CLI's and privileged profile password.

In a medium-security shop no user's profile allows both special local privileges (like Superuser) and network or modem privileges. A user with powerful local and remote privileges in one profile can roam the system from outside the site, browsing, reading, deleting, and placing Trojan horses at will. (A Trojan horse is a program designed to do something useful for a site — and does this — but also does things it was not authorized to do, like gather confidential information, such as passwords, for its author.)

Backup media is well organized — it may or may not be kept physically locked up.

Software that may represent a threat to security, like application programs, may be checked for possible trespass (detailed under the "The Closed Shop" section, next).

The Closed Shop

In a closed shop, few people have physical access to the computer, disk units, or system console. Users (including application programmers) work on terminals in a separate area. The CPU, disk and tape units, and even line printers are out of bounds to all but a few carefully screened and trained people. In closed shops, system operators are on duty most of the time. They mount and dismount tapes for users, handle printers, start application programs, do backups, bring the system up and down, and so on.

A closed shop should have someone check periodically, perhaps daily, for security violations and potential (and actual) break-ins. The site can run system logging with detail set to full, generate logging reports, and have the assigned person examine the reports for signs of unauthorized access.

Very few users in closed shops have Superuser or other special privileges. Any privilege that allows a user to bypass access controls means that he or she must be trusted; system security depends as much on privileged users as on operating system enforcement of access controls.

ACLs in closed shops are quite restrictive. When multiple users need access to a file, a closed shop may use explicit ACLs that spell out usernames, instead of using templates.

Software that's added to the core system, such as networking software, should be used very carefully in closed shops because it can compromise all security. For example, someone may check the files accessed by application programs for signs of Trojan horse incursions. When new versions of application programs are built, the new program file(s) or source file(s) may be compared to the old versions (using the FILCOM or SCOM program), and the differences examined. Media containing updates and revisions of Data General software should be checked, before being installed on the system, to make sure they are genuine Data General products with Data General labels, part numbers, etc.

Software acquired from an online bulletin board, via modem or otherwise, should not be added to the system unless the original source is known and reliable.

The *people* in a closed shop are essential to its security. Users can't touch disk and tape units, so at least one system operator stays on duty while the system runs. Administrators (system managers and operators) make decisions that can affect all access controls. They must plan and implement a secure system (using secure profiles, secure hardware, and ACLs) start logging, generate and check reports, plan user education, enforce password changes, and so on.

In some closed shops, system operators themselves have limited powers. A locked CLI runs on the system console, and very few people know the password. The operator must use EXEC and other CONTROL commands to run the system.

C2-Level Systems

One kind of closed shop is a *C2-level system*. The U.S. National Computer Security Center (NCSC) has defined several classes of computer systems from the standpoint of security; one of these is class C2. A system that meets C2-level security standards (if approved as such by the U.S. government) may be used, according to the rules for this class of systems, to handle sensitive and classified information. AOS/VS II Revision 3.00 is being evaluated for a C2 level of trust.

What Is a C2-Level System?

A C2-level system must meet minimum security requirements in the following general categories:

- *controlled access* to information
- *accountability* — identification of each user and tracking of his or her security-related actions on the system
- *assurance* of continued correct and secure system operation

The core of a C2-level shop is the software and hardware that people rely on to operate securely. This system core is called the *Trusted Computing Base (TCB)*.

For details of C2 and other U.S. government security standards, see the Department of Defense directive, *Security Requirements for Automatic Data Processing (ADP) Systems*, number 5200.28.

Components of the Trusted Computing Base (TCB)

For AOS/VS II Revision 3.00, software in the TCB includes the

- AOS/VS II kernel and tailored system (which enforce access controls)
- Peripheral manager (PMGR and its companion IACRS or CPIRS program). These programs manage character devices such as user terminals.
- PREDITOR profile editor, which creates and edits user profiles, and can provide password encryption
- EXEC and its companion programs XBAT, XLPT, XMNT, which oversee user logon and tape mount requests, and manage batch and printer processes. EXEC has Superuser privilege and can bypass access controls.
- Agent, which provides the user interface
- Other utility programs and files supplied with AOS/VS II, such as REPORT, FSCOPY, and DUMP_II/LOAD_II
- ADEX diagnostic system, to verify your hardware using the same microcode as AOS/VS II

Note that operating system access controls work *only* if, when the system was generated, the VSGEN parameter *Access* remained Y. The Access parameter is set to Y by default. If the Access parameter is changed to N, the resulting operating system will ignore access controls — it will be wide open. You can check all VSGEN settings, including the Access parameter, by typing TYPE SYSGEN:sys.CSF and pressing NEW LINE, where sys is the name of the AOS/VS system. Generally, unless your username is OP, you must turn Superuser on to read this file.

Before loading new AOS/VS or AOS/VS II software make sure the tape or diskette containing it was provided by Data General. It should be clearly labeled as a Data General product, with a Data General part number and copyright symbol.

Hardware in the TCB includes

- the system console (because anyone with access to it can penetrate all safeguards)
- ECLIPSE MV/Family CPU and microcode
- disk and tape controllers
- host–bus adapters
- disk and tape units and I/O storage systems
- all removable media (like backup tapes) that contain security-related information
- printers, if they print sensitive material and are accessible to users

C2 Configuration Hardware

Along with AOS/VS II Revision 3.00, the following Data General hardware is being evaluated by the NCSC for a C2 level of trust:

Processors

MV/1000 DC and MV/1000 RM
MV/1400 DC and MV/1400 RM
MV/2000 DC and MV/2000 RM
MV/2500 DC and MV/2500 RM
MV/3200 DC and MV/3200 RM
MV/3500 DC and MV/3500 RM
MV/3600 DC and MV/3600 RM
MV/4000
MV/4000 SC
MV/4000 DC
MV/5500 DC and MV/5500 RM
MV/5600 DC and MV/5600 RM
MV/7800
MV/7800 DC
MV/7800 DCX
MV/7800 XP
MV/8000 II
MV/8000 C
MV/9300
MV/9500
MV/9600
MV/10000
MV/10000 SX
MV/15000 Models 8, 10, and 20
MV/18000 Models 1 and 2 and MV/18000 SX
MV/20000 Models 1 and 2
MV/30000 Models 1, 2, 3, and 4
MV/35000 Models 1, 2, 3, 4, 5, and 6
MV/40000
MV/40000 HA Models 1, 2, 3, and 4
MV/60000 HA Models 1, 2, 3, and 4

Processor hardware includes processor boards, memory boards and system console.

BMC/DCH Controllers

Model Numbers	Description
4593	Dataproducts Parallel Printer
6795	Centronics Parallel Printer
6433	SCSI I Disk/Tape (Single–ended)
6434	SCSI I Disk Only (Differential)
6435	SCSI I Tape Only (Single–ended)
6786	SCSI II Disk/Tape (Differential)
6787	SCSI II Disk/Tape (Single ended)

MRC Controllers

Model Numbers	Description
80021/2	BMC E–MRC Channel Processor
80020	MV/40000 Channel Processor
80018/9	MRC System Interface
80013	MRC Bus Controller
80030	MRC RAMS Disk Controller
80023	MRC ARGUS Disk Controller
80033	MRC Tape Controller
6823	MRC SCSI II Disk/Tape (Differential)

Terminal Controllers

Model Numbers	Description
5916G	FCM/16
4543	MCP1
4359/67/69	IAC/8 (uECLIPSE)
4624/25	IAC/8 (68K)
4360/68/70	IAC/16
4622/23	IAC/24
5093LMC/4806	LMC/8
4814	LMC/8 II
4560	LAC/12
4712/13	LAC/16 II
4750/4803	LAC/16
4626/7/6C/7C	LAC/32
4626S/7S	LAC/32 II

Disk Drives

Model Numbers	Description
6067	50–Mbyte Removable
6060	96–Mbyte Removable
6061	192–Mbyte Removable
6122	277–Mbyte Removable
5061RSD	73–Mbyte Removable Winchester
5061RDD	146–Mbyte Removable Winchester
6627	590–Mbyte Removable Magneto–Optical
6670	332–Mbyte Removable SCSI
6671	662–Mbyte Removable SCSI
6030	370 Kbyte 8" Floppy
6096/97	1.2–Mbyte 8" Floppy
4514	48 TPI 5 1/4" Diskette
6309	96 TPI 5 1/4" Diskette
6098/9	12.5–Mbyte
6100/3	25–Mbyte
6225	5–Mbyte
6227	15–Mbyte
6234	50–Mbyte

6160	73–Mbyte
6161	147–Mbyte
6214	602–Mbyte
6236/7	354–Mbyte Argus
6239/40/90	592–Mbyte Argus
6357/98/99	862–Mbyte Argus
6581/2/4	500–Mbyte RAMS
6631/2/4	662–Mbyte RAMS
6621/2/4	1.2–Gbyte RAMS
6310	38–Mbyte ST506
6328	70–Mbyte ST506
6329	120–Mbyte ST506
6363	160–Mbyte ST506
6446	234–Mbyte SCSI
6491	322–Mbyte SCSI
6554	662–Mbyte SCSI
6492/6578/9	727–Mbyte SCSI
6716/6718	1.4–Gbyte SCSI
6539	179–Mbyte SCSI
6662	332–Mbyte SCSI
6796/6799	520–Mbyte SCSI
6685/6740	1.0–Gbyte SCSI
6841	2.0–Gbyte SCSI
7905	30–Disk SCSI Array
7907	20–Disk SCSI Array

Tape Drives

Model Numbers	Description
6026	800/1600 BPI 9–Track
6299/6300	1600/6250 BPI 9–Track
4307–TL	800/1600/6250 BPI 9–Track
5123SC/6341–A/6125	1600 BPI 9–Track
6586/7	1600 BPI 9–Track SCSI
6588/9	800/1600/6250 BPI 9–Track SCSI
6231/6311	15–Mbyte Cartridge
6351/6444	21–Mbyte Cartridge
6426	130–Mbyte Cartridge
5080	750–Mbyte Cartridge
6590	2–Gbyte 8mm Cartridge SCSI
6760	5–Gbyte 8mm Cartridge SCSI
6577	150–Mbyte QIC SCSI
6677	320–Mbyte QIC SCSI
6762	4–Gbyte 4mm DAT SCSI
7921	4–mm DAT Array

Terminals

Model Numbers	Description
6084/85/93	Hardcopy
6040/41/42	TP1 Hardcopy
6075/6193/6194	TP2 Hardcopy
6424/28/40	D577 System Console
6455	D578E System Console
6052/53/54/55	D1/D2/D3 Mono Alphanumeric
6182/242	D210 Mono Alphanumeric
6169/243	D211 Mono Alphanumeric
6344/91	D214 Mono Alphanumeric
6345/88/92/95	D215 Mono Alphanumeric
6500/05/20/65/66/21/78	D216/D216+/D216E/D216E+ Mono Alphanumeric
6682	D217 Mono Alphanumeric
6284	D220 Mono Alphanumeric
6692	D230C Color Alphanumeric
5654	D430C Color Alphanumeric
6166/255	D410 Mono Alphanumeric/Graphics
6346/89/93/96	D411 Mono Alphanumeric/Graphics
6501/22/67	D412/D412+ Mono Alphanumeric/Graphics
6683	D413 Mono Alphanumeric/Graphics
6167/256	D460 Mono Alphanumeric/Graphics
6347/90/94/97	D461 Mono Alphanumeric/Graphics
6502/04/23/24/68	D462/D462E/D462+ Mono Alphanumeric/Graphics
6684	D463 Mono Alphanumeric/Graphics
6291	D470C Color Alphanumeric/Graphics
6150	G300 Graphic
6241	G500 Graphic

Parallel Printers

Model Numbers	Description
4327/28	230/300 line–per–minute band printer
4363/64	436/600 line–per–minute band printer
4373/74	872/1200 line–per–minute band printer
4595	300 line–per–minute band printer
4596	600 line–per–minute band printer
4597	1200 line–per–minute band printer
4598/4603	1500 line–per–minute band printer
4599/4604	2000 line–per–minute band printer
6216	180 character–per–second dot matrix printer
4355	200 character–per–second dot matrix printer
6617	400 line–per–minute line dot matrix printer
6618	800 line–per–minute line dot matrix printer
6640T	9 page–per–minute laser printer
6646T/6779T	9 page–per–minute PostScript laser printer
4425	12 page–per–minute laser printer
6771	16 page–per–minute laser printer
6772/3	16 page–per–minute PostScript laser printer
6479	26 page–per–minute laser printer

Serial Printers

Model Numbers	Description
4433	150 character–per–second dot matrix printer
4434/51	160 character–per–second dot matrix printer
6215	180 character–per–second dot matrix printer
4535	200 character–per–second dot matrix printer
4589/90	240 character–per–second dot matrix printer
6594	400 character–per–second dot matrix printer
6425	300 character–per–second dot matrix printer
6647/48	300 character–per–second dot matrix printer
6788	300 character–per–second dot matrix printer
6514/5	300 character–per–second dot matrix printer
4354	340 character–per–second dot matrix printer
6789	622 character–per–second dot matrix printer
6617/18	800 line–per–minute line dot matrix printer
4518	35 character–per–second letter–quality printer
4320/22	55 character–per–second letter–quality printer
6321	40 character–per–second letter–quality printer
5431	50 character–per–second letter–quality printer
6640	6 page–per–minute laser printer
6646	6 page–per–minute PostScript laser printer
4557/8	8 page–per–minute laser printer
6454	8 page–per–minute laser printer
6640T	9 page–per–minute laser printer
6646T/6779T	9 page–per–minute PostScript laser printer
4424/26	12 page–per–minute laser printer
6474/5/6/7	12 page–per–minute laser printer
6480	12 page–per–minute PostScript laser printer
6771	16 page–per–minute laser printer
6772/3	16 page–per–minute PostScript laser printer
6479	26 page–per–minute laser printer

You will invalidate the C2 rating if you use any processor not named above or any of the items described in the next section.

Items Not Permitted in a C2-Level System

Certain types of hardware and software, while useful or even necessary in certain installations, are difficult to control from a security point of view, and hence are not permitted in a C2-level system.

- *Networking*, *virtual terminals*, and *modems* all expose a system to individuals who are physically remote from the site and possibly not part of the organization. These factors make it difficult or impossible to impose the controls and accountability recommended for all closed shops and required for a C2-level system. A knowledgeable person working from a remote location via networking or a modem may be able to gain access to the system and read, copy, change, or delete files without being subject to the normal access controls and user guidelines.
- *TermServers and TermControllers* all expose a system to individuals who may not be off site but who are outside of a department.
- *Custom logon and user cooperatives,* while supported by AOS/VS II, are not permitted in a C2–level system.
- *Shareware* or unpublished software tools and programs available from on–line bulletin boards, etc., may contaminate the system either intentionally or unintentionally and should not be included in a C2-level system.
- *Graphics terminals* and *windowing* are also not permitted on a C2-level system: to use graphics terminal capabilities a user must get certain file access privileges that threaten security. For details, see the section "Preventing Unauthorized Access to Windows."
- *Guest accounts* or usernames which several people can use may not be used on a C2-level system, because they violate the principle of accountability: that each individual user can be uniquely identified.
- Any application that alters the TCB.
- A connection that allows diagnostics to be run remotely.

How to Create a C2-Level System — A Summary

This section summarizes the steps involved in building and maintaining a C2-level system.

1. Format a system disk and install AOS/VS on it, and format other disks you may have, as described in *Installing, Starting, and Stopping AOS/VS*. These procedures are the same for C2- and non-C2-level systems.
2. Run VSGEN to create a tailored system as described in *Installing, Starting, and Stopping AOS/VS II*. Be sure to retain the default answer (Yes) to the Parameter question *Access?*. Patch the system, including any C2–specific patches (see the release notice for the names of the C2–specific patches), and test the system.
3. Create user profiles and start EXEC as described in *Installing, Starting, and Stopping AOS/VS II*. Do not give any users the following privileges:

   - Use virtual console (relates to networks, which aren't allowed with C2)
   - Access local resources from remote machines (also relates to networks)
   - Access devices
   - Change username
   - Superuser
   - Superprocess
   - System manager privilege
   - Modem

   Aside from C2 security issues, you can and should prevent users from denying service. You can prevent this by not granting the following privileges:

   - Create without block
   - Unlimited sons
   - Change priority
   - Change type
   - Change working set limit

   and by selecting the default answers for the *Priority* and *Max qpriority* questions.
4. Run LOCK_CLI or a locked 32–bit CLI on the system console, as described in Chapter 11. Before running it, change the password as described in this chapter. (The system console must be kept physically secure in a C2 system, but running LOCK_CLI or a locked 32–bit CLI provides an additional layer of security.)

5. When people start using the multiuser environment, maintain a system log as described in Chapter 11. Periodically, or whenever you're very concerned with security, you might want to run a detailed system log; then generate and read reports on security-related issues.

   As described in this chapter in the "Auditing with the System Log" section, outline a plan to create reports regularly (using the REPORT program) from your system logs. Someone will need to read the reports and check for suspicious events (described later in this chapter in the "Detecting Security Breaches" section).

6. Maintain a sense of security consciousness in your user community.

7. Keep your backup media secure, as described in this chapter in the section "Protecting the System Site and Backup Media."

8. Keep the computer, system console, disk units, and tape units physically secure. If the line printer(s) prints classified or sensitive material, keep it physically secure. Doing these things is described later in this chapter in the section "Protecting the System Site and Backup Media."

9. Do not run network software, remote terminals (via modems), or graphics (pixel-mapped) terminals on your system.

10. Follow other guidelines suggested in this chapter.

The following sections describe the components of the operating system that, either automatically or through proper system management, provide system security at the desired level. When necessary, the steps necessary for C2-level security will be specified.

Operating System Security Features

There are four security-related areas in data processing: discretionary access control, object reuse, identification and authentication, and auditing. Operating system security features in each area are as follows.

Discretionary Access Control

Discretionary access control means that access to both user files and system files is controlled by the person responsible for the file in each case. In an AOS/VS or AOS/VS II system, every file has an access control list (ACL), created automatically when the file is created. Only those users explicitly given file access by the ACL can read it or perform any other action with respect to it. (The exception is users with the Superuser privilege, who have full access to all user and system files.)

For a file created by a user, the initial ACL is the user's default ACL, which, if unchanged, prevents access by anyone other than this user (except for a superuser). *System* files are shipped with ACLs that protect them from all users except superusers and the system operator (username OP). ACLs are described in detail in the section "Controlling Access with ACLs," and also in Chapter 3.

Object Reuse

The words "subject" and "object" provide a common ground in the context of computer security. A *subject* is an active entity that causes information to flow among objects or changes system state. Examples of subjects include users, processes, and system calls in running programs.

Objects are passive entities that contain or receive information. Objects include both physical devices such as printers, video displays and keyboards, and entities such as records, blocks, pages, files and directories. A *storage object* is one that supports both read and write access — for example, a disk block or memory page — and hence can be reused, when its data is no longer needed, to store new information. For a system to be secure, all information in a storage object must be removed before the object is reused — as in the zeroing of page frames, for example.

Deleting a file frees the data blocks allocated to the file. The operating system zeroes these blocks *before* it allocates them to any other file. This prevents users from creating new files, using positioning calls to allocate blocks that may contain data, and then reading the blocks.

The operating system manages main memory pages much the same way as it does files. Memory pages used by a program are not zeroed when the program terminates or when it's paged, but the page frames *are* zeroed before being assigned to another program. (Specifics on memory-page reuse appear in the manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary*, under calls ?ALLOC, ?ISEND, ?IS.R, ?MEMI, ?SPAGE, ?READ, ?WRITE, ?RDB/?WRB, and ?PRDB/?PWRB.)

In these ways, the operating system prevents users from retrieving sensitive information through object reuse.

Identification and Authentication

With identification and authentication the system restricts access to only authorized users, protects the privacy of each user's files, and links each user with his or her actions on the system. This is made possible by the username-password combination.

Every system user is assigned a unique username and a password as part of his or her user *profile*. To log on, a person must have a valid user profile (valid account); and must type the username and password stored in the profile file. Passwords are 6 to 15 characters long and can — at the system manager's discretion — be encrypted.

All user profiles are stored in directory :UPD and have null access control lists (ACLs), which means that only privileged users (superusers) can read or change them. Profiles are created, changed, and deleted via PREDITOR, the user profile editor. PREDITOR requires Superuser privilege to run, which prevents any but a superuser from creating, examining, or changing *any* profile. The PREDITOR defaults are null (do not grant privilege) for questions about Superuser and other special privileges.

While a user is logged on, his or her username remains constant. The ability to change a username requires a special privilege, which should never be given to a user under normal circumstances. The privilege to change password, on the other hand, is normally given to each user; changing password from time to time is one good way to protect the privacy of one's files.

The username/password arrangement enables the operating system to identify and authenticate users.

User privileges and security are explained later in this chapter. Details on PREDITOR appear in Chapter 2.

Auditing

Auditing refers to the ability of the system to keep a record (an "audit trail") of specified events, as the system manager considers necessary for system security. At the discretion of the system manager, the operating system can run a detailed system log, which records each file access and other security-related event. From a detailed log, the REPORT program can create reports based on username, filename, or failed logon. For each event, the report includes error information (if access failed); the absence of error information means that access succeeded.

The ability to audit users and events can help security-conscious sites determine whether someone has tried to, or actually *has*, broken into a system. The ACL of the system log file (:SYSLOG) is null; you must have Superuser privilege to read it, generate a report from it, delete it or modify it.

Detailed logging and REPORT program operations are explained in Chapter 11.

The following sections describe the procedures to be followed by system managers, operators, and users to establish and maintain system security at the desired level.

User Privileges and Security

The process created for each user at logon has only those privileges specified in the user's profile. Unless a user has special privileges, the operating system will reject commands that might threaten system security. For example, if a user tries to type a file he or she's not authorized to read, the system will respond with an *Access denied* message; and if a user who lacks Superuser privilege tries to turn Superuser mode on, the system will respond *Caller not privileged for this action.* Thus, the primary basis for a secure system is user profiles, created by the PREDITOR profile editor. Your system is only as secure as its users' profiles.

Table 14-2 names and describes privileges that can protect or threaten system security. The privilege questions appear in the order asked by PREDITOR. Details on all privileges appear in Chapter 2.

Table 14-2 User Profile Privileges that Relate to Security

Privilege	What It Allows
Encrypt password	Without this privilege, a user's password is stored in the :UPD file in such a way that a superuser may be able to see it by reading the profile file. Encrypting the password makes it impossible to read. This is very desirable from a security viewpoint.
Initial IPC file	This value determines which file (if any) will be executed for the user at logon. The IPC file can be a macro that sets default ACL (DEFACL), search list, and other things. For greatest control, we suggest you use a central logon macro (perhaps in :UTIL) to execute individual user logon macros. Give the central macro's name as the initial IPC file.
Use Console Use Batch	Either of these privileges allows a user to gain entry to the system. To deny the Use Console privilege is torestrict the user to using a card reader.
Use virtual console Access local resources from remote machines	These values determine whether the user will be able to log on to your system, use a terminal connected by means of a soft controller (like PAD or TELNET) or use resources on your system from a remote system over a XODIAC network. A privileged user (one with (the Superuser or Superprocess privilege) should *not* have either network privilege, since he/she could then explore your system at leisure from a remote site. If you want a user to have both network and either Superuser or Superprocess privileges, create two profiles for the user, one profile with network privileges, the other with the Superuser or Superprocess privilege.

(continued)

Table 14-2 User Profile Privileges that Relate to Security

Privilege	What It Allows
	Because of the security risks involved, networking is not permitted, and these privileges would not be assigned, in a C2-level system.
Use IPC	This privilege allows a user process to send Interprocess Communication (IPC) calls to any process in the system. CEO users need this privilege, and other programs may need it. Since the operating system logs IPC calls, granting this privilege is not a breach of security.
Create without block	A Yes value allows the user process to create sons (that in turn can create sons) without blocking. All its sons (and grandsons) can remain unblocked, increasing system overhead. This privilege relates to the next two. A CEO user needs this privilege.
Change Password	This privilege allows users to change their password while logged on. Because users should change their passwords often in a secure system, you should give this privilege to users.
Unlimited sons Sons	These two values control the number of sons the user process can create. Generally, a user should not have the unlimited sons privilege. Each sonrequires some system overhead, especially if its father isn't blocked (above). Five sons is enough for practically anyone. Some CEO users need 12.
Change priority	The user process (and sons) can change its own priority if it has this privilege. It can gain favor over other processes; thus restricting access to others.
Change type	The user process (and sons) can change type (for example, to resident) if it has this privilege — gaining favor over other processes.
Change username	The user process can create a son process that has a different username. The ability to change username allows the user to *become* any other user, and thus gain access to the other user's files. With the username OP, a user can gain unlimited access to many system files — since, to help the system operator manage, these are shipped with ACLs that give OP unlimited access. Also, with the username OP, the user can issue EXEC commands (allowing the user to terminate user processes or change priority or type). Also, any user with this privilege can't be traced; he or she can try to break security under another person's username.

(continued)

Table 14-2 User Profile Privileges that Relate to Security

Privilege	What It Allows
Change username (continued)	For user files to remain secure, and logging to be useful, every user on your system should have his/her unique username — and be unable to change it. Do not give users this privilege if you want a C2-level or other secure system.
Access devices	The user process can define a new device (call ?IDEF), wire pages, access the device, and/or remove the definition. For this privilege to work, the process must be resident, which means the user must also have the *Change type* privilege. These privileges give the user *at least* the power to dominate or bring down the system. In a C2-level system, users are not given this privilege.
Superuser	Superusers can bypass all file access controls. They can run PREDITOR to give themselves any privilege — or possibly learn other users' passwords (if you don't choose encryption). Do not give users Superuser privilege if you want a secure system.
Superprocess	The user process (and sons) can change its own type and priority; and it can block or terminate any other process, including LOCK_CLI. Do not give users this privilege if you want a secure system.
System manager privilege	This privilege allows the user to initialize and release job processors (if your computer has more than one job processor), create and delete classes and logical processors (usually done via the optional CLASP utility), and change the locality of other users' processes. System Manager privilege also allows a user process to issue system calls that change the system date, time, ID (SYSID), and bias factor. Also, the user can start or stop the system log (SYSLOG). These privileges have significant impact on security. Use of classes and privileged system calls can affect the performance and security of your system. Generally, the master CLI issues all commands that require system manager privilege; do not give it to any other user unless he or she really needs it.
Modem	A Yes value allows the user to log on your system from a modem (over a phone line). You may want to allow selected people modem privilege, with certain precautions against break-ins on the line (described later).

(continued)

Table 14-2 User Profile Privileges that Relate to Security

Privilege	What It Allows
Modem (continued)	A privileged user (one with the Superuser or Superprocess privilege) should *not* have modem access, since he/she could then explore your system at leisure from a remote site. If you want a user to have both modem and either Superuser or Superprocess privileges, create two profiles — one with each privilege — for the user. Modems, like networking, are not allowed in C2-level secure systems.
Change working set limit	A Yes value allows the user process (and sons) to override the working set limit that the operating system dynamically assigns to each process. This privilege is needed for any of the Logical address space and Minimum/Maximum privileges below. Changing the working set size may allow the user process to monopolize memory and the system.
Priority	If given a higher-than-default priority, the user process (and sons) will get CPU preference over processes with lower priority. This privilege doesn't allow the user to *change* priority (see above).
Logical address space - batch Logical address space - non-batch	These parameters give the user process (and sons) a nondefault number of memory pages in batch or interactive processing. The default is a very high number.Giving a nondefault value might have some effect on system performance.
Minimum working set size–batch Maximum working set size–batch Minimum working set size–non-batch Maximum working setsize - non-batch	These parameters give the user process (and sons) specific minimum and maximum working set limits in batch or interactive situations. Giving a nondefault value can have significant effect (usually bad) on performance, depending on your application's environment.

(concluded)

Privileges in a C2-Level System

As explained in Table 14-2 above, C2-level system users would *not* be given the following privileges in their profiles:

- Use virtual console
- Access local resources from remote machines
- Change username
- Access devices
- Superuser, Superprocess, or System Manager
- Modem

Process Privileges

Some *processes* automatically have special privileges, regardless of those specified in any user profile. These are listed in Table 14-3.

Table 14-3 Privileged Processes

Process	Privileges
PMGR (peripheral manager) PID 1, username PMGR	The peripheral manager has all possible privileges. It is run by the operating system kernel and is not accessible to users.
Master CLI, PID 2, username OP	The initial CLI process runs on the system console. (You can select a different master process during VSGEN, if you want.) The master CLI is the progenitor of all other processes and has all privileges. (Also, as PID 2, it can set the system date and time, start or stop the log file, and set default characteristics for user terminals; no other process has these powers.) Processes created by the master CLI via the XEQ or EXECUTE command have all the master's privileges. Processes created by the master CLI via the PROCESS command have no special privileges unless the PROCESS command specifies privileges with switches. You can't remove the master CLI's privileges but can run a locked CLI under it — making some of its powers unavailable to users. Running a locked CLI is described later in this chapter.
EXEC, PID 3 (usually), username EXEC	EXEC is created by the master CLI with all privileges. It creates user processes with the privileges specified in the user profiles. EXEC commands relate to the batch, spooling, printers, tape units and tape mounts, and the multiuser environment in general. Any process with EXEC's father's username (usually OP) can issue commands to EXEC.
Any process, with any PID, username OP	Any process with username OP has owner access to many system files. (These files are shipped with ACLs of OP,OWARE to help the system operator run the system without turning Superuser on.) Also, any process with EXEC's father's username (usually OP) can issue commands to EXEC — giving it immense powers (although it cannot issue privileged CLI commands, like SUPERUSER ON). By default, any CLI run by the master CLI has username OP. This means that a locked CLI, unless it was created with a username other than OP, can issue EXEC commands.

Protecting the system console is described later in this chapter.

Logon Procedures and Security

Before logon, the username-password pair is the first — and sometimes only — line of defense from break-in. After logon, there are sometimes signs of break-in that users can notice. This section tells you how to maintain the strength of the logon defense and how users can help protect themselves.

NOTE: The security procedures described in this section will have no effect if users leave their terminals unattended while logged on.

Logon Tries

To log on, a user must type his/her username and password correctly. By default, EXEC allows five logon tries (which may be break-in attempts). After five unsuccessful logon tries, EXEC locks the terminal for 10 seconds (local line) or breaks the connection (modem line). The terminal lock and broken connection are designed to discourage break-ins.

In the ENABLE command that activates each terminal line, you can specify a different number of logon tries allowed — from 1 to 10. And, you can tell EXEC to disable the line (instead of locking it for 10 seconds or breaking the connection) after a given number of logon tries. A disabled line can't be used again until someone at the system console re-enables it using the EXEC command ENABLE.

For greatest security, you might allow only one or two tries, and tell EXEC to stop (disable) the line thereafter. This is particularly relevant on modem lines (if you have them), since anyone who has a Data General-compatible terminal and modem, and who knows your system's phone number can try to guess a username/password pair and — if she or he succeeds — break into the system. Many people who've used AOS/VS or AOS/VS II know that each system has an OP profile and that (often) the OP profile is privileged. All they need to do is guess the OP password (this is a good reason to keep modem, Superuser, and Superprocess privileges in separate profiles).

EXEC's ENABLE command, which allows you to change the number of tries allowed on a line, and specify what will happen then, is described in Chapter 3.

Logon Banner

The system logon banner (changeable via the SYSID command) identifies your system to users before they log on. You can edit it to identify your system uniquely. However, there is some measure of security in anonymity; therefore you may want to retain the anonymous default banner, which says only "AOS/VS [II] n.nn" where n.nn is the revision number.

This is especially pertinent if you have a number of modem lines, since the default system ID gives little information to anyone seeking to break into your system over the telephone. (Modem lines themselves are not part of the C2 Trusted Computing Base, and shouldn't be part of a secure system; but you may have compelling reasons for using them.)

User Logon and Logoff Messages

Logon and logoff messages can help you and users identify possible system break–ins.

Whenever a user logs on or submits a batch job, the following things happen:

1. A user types username and password, or the system accepts a batch job. Users can post batch jobs with the command QBATCH or QSUBMIT. Normally, a batch job is processed as soon as possible — but, if desired, a user can tell the system to run the job *after* a specific time, using the QBATCH or QSUBMIT switch /AFTER=.
2. EXEC verifies the username and password, and tells the system to create the user process.
3. The system creates the user process for the user, running the initial program specified in the user profile.
4. The system writes the greeting message (LOGON.MESSAGE) to the user.
5. The system tells the initial program to execute the initial IPC file specified in the profile.
6. The system notes the date and time in the user's profile file (in :UPD) and in the system log (if logging is on).

The process then runs for the user until he or she logs off or the batch job terminates. Then, the following things happen:

1. The user logs off or the batch job terminates.
2. The system terminates the user process.
3. The system notes the date/time in the user's profile file (overwriting the logon time) and in the log file (if any).
4. The system writes a process termination message to the user. If the user was logged on over a modem line, EXEC automatically hangs up, breaking the connection.

During logon and logoff, messages are written to the terminal. For batch, messages are written to the batch output file (by default, the line printer).

Logon and logoff messages can help you, and help users, identify possible system break-ins. Encourage users to check the displayed date/times each time they log on. If the displayed date/time doesn't match the last time the user logged on, this means someone else has been using this user's account. The user should change his/her password immediately and tell you (or tell the security person, if it isn't you) about the breach.

Sample system logon and logoff sequences follow.

First, the user types username and presses NEW LINE, and then the password and NEW LINE; then the system displays

---

Last message change *2-Oct-93* *15:32:08*

... (System displays contents of file LOGON.MESSAGE) ...

Last previous logon *3-Oct–93* *8:22:59*

... (System executes commands in user logon macro, if any) ...

AOS/VS II CLI Release n *4–Oct–93* *8:23:01*
)

User works with system until lunch time and then logs off using the BYE command and pressing NEW LINE.

) BYE ↵
AOS/VS II CLI TERMINATING *4–Oct–93* *12:04:44*

Process n Terminated
Connect time 3:31:43
User 'xxx' Logged off @CONn *4–Oct–93* *12:04:47*

If another user process with the same username is still running, the termination (obituary) message looks like this:

Process n Terminated
Connect time hh:mm:ss (the batch message is *Connect time hh:mm:ss)*

(Other jobs, same username:)

Number of console jobs, n
Number of batch jobs, n

User 'xxx' Logged off dd-mon-yy hh:mm:ss

Ask users, when they log off, to

- check for an *(Other jobs, same username)* message. This message means the system is running another process for the same username. By comparing the number of console (terminal) and/or batch jobs given in the message, with his or her actual number (if any) of current batch jobs or logons at other terminals, the user can tell if someone else is using his or her username. If this seems to be the case, the user should tell you at once.

Ask users, when they log on, to

- check the *Last previous logon* date and time, as described above. If this doesn't match the time they remember, someone else may have been using the account. The user should tell you at once.

User Passwords

The person who creates a user profile also assigns the user's password. The password must be between 6 and 15 characters long. Case of letters is ignored. More than 15 characters won't work — if the password is 15 characters, and the user types these 15 and one more, the system will reject the password.

A user's password can include any printable character. Obviously, a password like !*%EO(~ is hard to guess (and, perhaps, to remember).

You can tell PREDITOR to encrypt a user's password before storing it. From a security standpoint, this is desirable because no one — not even a superuser — can figure out an encrypted password. It is a good idea to encrypt passwords. Encryption will *not* affect access by means of CEO Mail to remote hosts or CEO printing on remote printers, nor will it affect access by means of TCP/IP or XODIAC/XTS network software. Remember that connection to a network is a security risk; do not participate in a network if security is critically important to you. Networks aren't allowed in C2-level systems.

Anyone who edits the profile can change the existing password without knowing the old one.

Users may be allowed to change their passwords. By default, PREDITOR grants this privilege (the default answer allows it).

Generally, users should be able to change their passwords, since if a user originates a password, he/she is likely to remember it. If someone else originates the password, the user is likely to forget it — and will probably write it down, producing a security risk.

Unfortunately, given free choice, users are likely to choose obvious passwords. Passwords that users should avoid (because they are easy to guess) are

- initials or part of name (particularly the username, which is very easy for people to try);
- name of housemate, spouse, child, or pet;
- make or model of favorite auto;
- name, make, or model of boat;
- phone number;
- name associated with work, like company, group, or project; and
- anything people know this user cares about.

Good password choices include a mixture of letters, numbers, and punctuation; for example

WLS1.1

One way to build good, memorable passwords is to create them from phonetic syllables. For example, take the syllables

ang fof lal dan pok soo

There are many combinations of phonetic syllables are easy to remember, yet obscure enough not even to exist in the dictionary.

Another possibility is a foreign language, if you know it well. Avoid obvious words or phrases.

Yet another possibility for a password is a phrase, with words separated by periods or underscores. For example,

save.my.files

Passwords should be changed regularly — depending on the sensitivity of a user's data. They should be changed *immediately* if you suspect your system is under attack (signs are described later). Also, passwords should be changed whenever a privileged user plans to leave, or leaves, your organization. This is true because a person who leaves a group also leaves group rules and sanctions — in theory, he or she has nothing to lose by taking information.

From time to time, users will forget their passwords. You can assign a new one via PREDITOR, if you edit the user profile. You need not know the original password to create a new password.

You can monitor user password changes with detailed logging.

Guest Accounts and Shared Passwords

A guest account, which allows users without individual accounts to log on to your system, is a useful communications aid. Such profiles have public usernames and passwords which may be the same; for example $GUEST and $GUEST. Guest accounts are useful because they allow people without accounts to read general-distribution bulletins and mail.

Unfortunately, guest accounts represent holes in security, because — since many people know the password — it's difficult to identify an account abuser. You can trace suspicious activity only to the username, not to a person. This is a problem with *any* account that two or more people share.

If security is critically important to you (closed shop), avoid creating a guest account. Guest accounts are not permitted on a C2-level system. If you really need a guest account, give it a username that has a unique character (for example, $ in $GUEST), so you can specifically *deny* access to this username in other users' default ACLs. For example, the central logon macro could include the command line

```
DEFACL [!USERNAME],OWARE $+,, +,RE
```

This would give the user ownership (as usual), anyone whose username began with $ (like $GUEST) *no* access, and other users read and execute access. Some files, like SED.PR in :UTIL, and mail files, must be accessible to users — access Read or Execute, or Read and Execute. Log on as a guest and try the operations you want guests to be able to do.

Avoid giving the guest account any of the privileges shown in Table 14-2 (except *Use console*). You might want to restrict the number of sons (to 0 or 1) to conserve system resources. And, you might restrict disk space to a reasonable minimum — say 500 or 1000 blocks.

Password-Stealing Programs

Any terminal left unattended (whether or not a user is logged on) is vulnerable to a password-stealing program or CLI macro (sometimes called a spoofer or password-grabber). Such a program usually involves a fake system logon banner, which makes it appear as if a terminal is unused when actually it is running the password-stealing program.

In one scenario, the program author runs the stealing program on an unused terminal, using a public account to avoid identification. Eventually, a user tries to log on, typing username and password. The program stores them in a file that its author can access (in some public directory). Then, the program displays *Invalid username-password pair* and terminates. The system displays *Process terminated* and *User 'xxx' Logged off*, identifying the username. The giveaway here is the *terminated* and *Logged off* messages, which never appear in a real logon sequence.

A stealing program run from a user process (a terminal in use left unattended) is more subtle. The original user returns to find the system banner on his/her terminal. He/she may not remember logging off, but blames the inconsistency on poor memory. He/she types username and password, and the program copies username and password as before; but then it chains to the CLI. It can produce a logon message and CLI banner identical to the original. The only clue is the *Connect time* message displayed when the user logs off. This message will show a longer connect time than expected, since the user's initial process was running the whole time.

There is no real defense against password-stealing programs — there's only detection. User awareness is needed to detect the first type of stealing program (*terminated* message). User consistency can prevent the second type of stealer: users who always log off when leaving their terminals won't be fooled.

When you, or a user, thinks a password might have been stolen, the user must change that password immediately.

The log file, with detail set to full, yields evidence of the second type of password stealing: a report of the original username's file accesses will show the steal file pathname.

Generally, you need the cooperation of users in many ways to run a secure system. For users, security features of both AOS/VS and AOS/VS II are described in the manual *Learning to Use Your AOS/VS System*. You might encourage users to read that book.

System Calls and Tailored Operating System Applications

The following information on system calls is provided for those who are programming special operating system-based applications. (For more information on system calls, see the manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary.*)

Terminal Access Control Rules

The operating system identifies and authenticates users during its logon procedure, as follows. When a user at a terminal starts to log on, the EXEC program asks for username and password. If the user types a recognized username-password pair, EXEC tells the system to create a process for that user on that terminal. The system creates the process with the privileges specified in the user profile. The user process has control of the terminal until the user logs off.

If you want to write a program to identify and authenticate users (replace EXEC for these purposes), the program must assign the terminal and open it (using privilege settings given in the user profile, pathname :UPD:username) according to the following rules.

Terminal files in :PER (CONn files, where n is a number) have ACL PMGR,OWARE. This ACL is not meaningful to PMGR access rules.

When the system comes up, all terminals are unowned.

Usually EXEC will gain ownership of all terminals through the EXEC ENABLE command.

To have ownership of a terminal you must assign it by either:

- explicit assign (system call ?ASSIGN);
- open assignment (system call ?OPEN); or
- process assignment (system call ?PROC)

Once you own the terminal, you may perform I/O after you open the terminal (?OPEN call) for I/O access.

If you own the terminal via ?PROC, this is your "process console" and you are the interruptible owner. (Interruptible ownership means that CTRL-C sequences affect your process.)

Your I/O access to a terminal is lost when your program closes the terminal (?CLOSE system call) or terminates.

Ownership and interruptible ownership are lost upon termination, deassignment, or passing the console to a son process (not shared). When a console is passed to a son process, the father process becomes suspended. The father regains the ownership when the son process terminates.

If an interruptible owner turns on the shared characteristic and passes the console to a son process, the father process remains the interruptible owner.

Request Types

Only a process with I/O access to a console may issue ?READ and ?WRITE.

Only the owner of a console may issue control requests (e.g., ?OPEN, ?GCHR, ?SCHR, ?STOM, etc.) with the following exceptions:

- PID 2, a process with System Manager privilege turned on, or the console owner may get the default characteristics
- PID 2, a process with System Manager privilege turned on, or the console owner may issue the ?CLRDV or ?CONINFO system calls

Any process may issue a ?SEND call to any process console (no access checking). However, a process can turn off ?SEND message receipt via a characteristic. This characteristic cannot suppress messages from PID 2.

The only control requests allowed on an unowned console are ?ASSIGN, ?OPEN, or ?PROC and the following exception: only PID 2 or a process with System Manager privilege turned on can set default characteristics, and can set them on any console.

Discretionary Access Control

The system calls that affect discretionary access control are

- get or set ACL (?SACL);
- get or set default ACL (?DACL);
- get or set Superuser, Superprocess, or System Manager mode (?SYSPRIV);
- manage group lists (?GROUP).

For specific information on the security requirements for these and other system calls, see the section "Who Can Use It" for each system call in the *manual AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary.*

Controlling Access with ACLs

ACLs are an important part of system security. The following sections offer some detail about ACLs and describe two ways to combine file sharing with system security.

In a multiuser system, users need access to, and privacy for, their own files. System and user files need protection from accidental or malicious deletion. Operating system file access control lists (ACLs) help ensure this.

NOTE: Access controls work *only* if, when the system was generated, the VSGEN parameter *Access* remained Y. The Access parameter is set to Y by default. If the Access parameter is changed to N, the resulting operating system will ignore access controls — it will be wide open.

Each directory, including the root (:), and each nondirectory file, has an access control list (ACL). An ACL is a list of matched username-access pairs, with the form

username, access *[username, access]* ...

A username can be a real username, or it can include filename template characters to specify a group of usernames. Access types are O (Owner), W (Write), A (Append), R (Read), E (Execute), and null. They have the following meanings.

Access Type	Allows
O (Owner)	Owner access to a directory allows a user to change directory ACL, rename, or delete the directory, regardless of the ACLs of files in it. The user can also change the ACL of any file — directory or nondirectory — within the directory. In addition to Owner, Execute access is needed if the user wants to enter the directory or use its name in a pathname. Owner access to a logical disk unit (LDU, set by the AOS/VS Disk Formatter utility and the AOS/VS II Disk Jockey utility) is needed for a user to initialize the LDU. Owner access to a nondirectory file allows a user to change ACL, rename or delete the file, check file status, check or change permanence, or create, read, edit, or delete a user data area (UDA) for the file. For a directory or nondirectory file, Owner access differs from Write access as follows: with Owner access, a user can delete the file *without having Write access to its parent directory*; the user needs only Execute access to the parent directory. With Write (but not Owner access), a user can't delete the file unless he/she has Write or Owner access to the parent directory. This means that if you want a user to be able to add to (but not rename or delete) a file in a directory to which the user has only Execute (or Read and Execute) access, give the file an ACL of username,WARE (not username,OWARE).

Access Type	Allows
W (Write)	Write access to a directory allows a user to delete or rename any file (including directories) in the directory — regardless of the file's ACL. To delete or rename the directory itself, a user needs Owner access to it or Write access to its parent directory. Write access to a directory also allows a user to create files andchange file ACLs there. Write access allows a user to initialize and release an LDU (the user also needs Owner access to the LDU and Execute access to the disk entry in :PER). Write access to a directory has little value by itself. To do anything in the directory, a user also needs Execute access to the directory. Write access to a nondirectory file allows a user to change file contents (for example, by editing with a disk file editor or adding text from another file), get a complete pathname, check file status, check or change permanence, and create or edit a user data area (UDA). Write access has little value without Read access, since without Read access, the user can't tell what's in the file. Generally, a user can't modify a file with a text editor unless he or she is able to delete it (he or she has Write access to the parent directory). This is true because at the end of the editing session the text editor tries to delete the original file and replace it with a new, updated file.
A (Append)	Append access to a directory allows a user to add files (directory or nondirectory) to it. Append access to a directory has little value by itself. To do anything in the directory, a user also needs Execute access to the directory. Append access to a nondirectory file allows a user to check the file status, get a complete pathname, and check or change permanence for the file.
R (Read)	Read access to a directory allows a user to read filenames in it (for example, in a FILESTATUS command, with or without templates). Read access also allows you to read ACLs of files *in* the directory. Read access to a directory has little value by itself. To do anything in the directory, a user also needs Execute access to the directory. Read access to a nondirectory file allows a user to read it — for example, with the TYPE command or a text editor. The user can also check the file status, get a complete pathname, and check or change permanence. The user also needs at least Execute access to the parent directory and all higher directories.

Access Type	Allows
R (Read) continued	Read access allows a user to copy a file or debug a program. This is *not* desirable for a program file, since a user can copy the file to his or her own directory and explore it at will (via a debugger or DISPLAY program). Or, he or she can use it to corrupt existing programs (for example, use a compiler to create a devious object file that will be linked into an application program).
E (Execute)	Execute access to a directory allows a user to enter the directory and use its name in a pathname. To list files or check file ACLs in the directory, a user must have both Execute and Read access. (Users *can* list a specific filename, if they know it, with Execute access only.) Execute access to a nondirectory file allows the user to execute thefile as a program (XEQ or PROCESS command), to read file status, get a complete pathname, and to check or change permanence. Execute access to a program file allows the user to run the program via the XEQ or PROCESS commands. To debug the program or run a disk file editor on it, the user needs Read *and* Execute access. Since Execute access alone doesn't allow the user to see the contents of the program, you can give Execute access safely to programs that (internally) contain sensitive information.
,, (null)	Null access excludes a user from access. For a directory, null access prevents the user from accessing files within the directory. The user can list the directory name. For a nondirectory file, null access prevents the user from accessing the file. By default, users who are not given access have null access anyway, which implies that you don't need to specify null access. But if you use a template to give a group of users access, null lets you *exclude* users in the group. You can assign null access specifically with ,, (two commas) after the username or template. NOTE: If you end an ACL command line with null, use one comma instead of two (e.g, ACL FILEA A+,WARE $+,). For example, assume you want to retain ownership of a file and give most users Read and Execute access to the file. You want to exclude users ANDY_B and any user whose username begins with $. You could type) ACL pathname [!USERNAME],OWARE ANDY_B,, $+,, +,RE ↵ To deny everyone (including its creator) access to a file, use the ACL /K switch, as follows:) ACL/K pathname ↵

When the operating system starts up, it assigns an ACL of +,E to the root directory, to allow users access to their own directories. This +,E ACL remains in force — regardless of any ACL assigned with the Disk Formatter or with the ACL command — while this AOS/VS system is running.

PREDITOR creates each user directory (in :UDD) with an ACL of username,OWARE. Each file the user creates within this directory gets the same ACL (username,OWARE) by default. The user profile files, in :UPD, have null ACLs, which means that only superusers, like EXEC, can read them. For reasons of security, do not change the ACL of :UPD or profiles in it.

Every user has Owner access to his/her files, and thus can change file ACLs with the ACL command, or change the default ACL with the DEFACL command.

A user can give another user (or all users) read access to a file by giving Execute access to his directories (up to :UDD:username) and by giving Read access to the file. For example, assume Sam wants to give Al read access to file :UDD:SAM:MARK_IV:MARKET_PLAN. Sam can type

```
) ACL :UDD:SAM SAM,OWARE AL,E ↵
) ACL :UDD:SAM:MARK_IV SAM,OWARE AL,E ↵
) ACL :UDD:SAM:MARK_IV:MARKET_PLAN SAM,OWARE AL,R ↵
```

(The SAM,OWARE is needed to retain Sam's access rights to his own file.)

If a user will often need a tailored, nondefault ACL, you or the user can specify this ACL in the user's logon macro with the DEFACL command. For example, assume that Sam will periodically want to give Al read access to selected files. Sam can put the command line

```
DEFACL SAM,OWARE AL,E
```

in his logon macro. This will allow Al to enter Sam's directories (but not browse for files in them). Sam can then let Al read a file by typing only *one* ACL command, regardless of the directory that holds the file; for example

```
) ACL :UDD:SAM:MARK_XV:STRATEGY SAM,OWARE AL,R ↵
```

In a more relaxed situation, you might want to specify Execute access for *everyone*. For this, the easiest method is to put the DEFACL command line in the *central* logon macro (that executes user logon macros). The command would be

```
DEFACL [!USERNAME],OWARE +,E
```

The DEFACL command can be inserted in a logon macro with a text editor.

The ACL structure makes it very easy to restrict access. All you need do is remove general Execute access from any directory above the one you want to restrict. For example, changing ACL of user Jan's initial user directory to JAN,OWARE prevents anyone other than Jan without Superuser from accessing any of her files — regardless of the ACLs below directory :UDD:JAN.

Username Groups in AOS/VS and AOS/VS II

If you want to group a set of users, create a username group. In AOS/VS and AOS/VS II, a username group is a set of users (usernames) that, combined with ACLs, allows (or restricts) access to a group of users.

To start, you need a unique specifier to use when creating the group members' user profiles with PREDITOR. Then, you can use this unique specifier in all pertinent users' default ACL settings (in logon macros) to establish the needed access to files.

For example, assume a project called Mark II that involves Sally, Jack, Alexis, and Sam. Sam is the project leader. You want to create a username group that gives each person execute and read access to other group member directories. In addition, you want Sam to have append access to group member directories. As a group name, you can choose something like +.MK2 (the suffix .MK2 is the unique specifier). Using PREDITOR, you create the profiles

```
SALLY.MK2 JACK.MK2 ALEXIS.MK2 SAM.MK2
```

Then, you create a logon macro for each user's directory. It contains the command

```
DEFACL [!USERNAME,OWARE] SAM.MK2,ARE +.MK2,RE
```

The group's default ACL gives each user OWARE access to his or her files; it gives Sam Append, Read, and Execute access to all group member files; it gives group members other than Sam Read and Execute access to all group member files; and it gives no access to other users. Since each user has Write access, he or she can change ACLs (including the default ACL) at will. You rely on the people in the project to retain the ACLs needed for communication.

After deciding on a unique group identifier, you can check the users that belong to it with either the FILESTATUS command in :UDD or PREDITOR templates. To see which users belong to the Mark II group, you could type

```
) SUPERUSER ON ⤵
```

```
Su) FILES/S :UDD:+.MK2 ⤵
```

... (it displays a sorted list of usernames)...

```
Su)
```

Username groups can improve the flow of information, while retaining tight access control; they also make it easy to clean up (by deleting accounts) after a project is completed.

The use of username groups is not permitted in a C2-level system.

Order of Usernames in ACLs

When you (or anyone) give access privileges to different users, be sure to assign *specific* username groups before *general* username groups (templates). For example, the ACL

```
+,RE SALLY,OWARE
```

gives all users (which includes Sally) Read and Execute access — and *only* Read and Execute access. The Owner, Write, and Append access for Sally are ignored. With this ACL, no one but a superuser can write to the file or change the ACL. But transposing the user groups:

```
SALLY,OWARE +,RE
```

gives Sally all access, and gives all users Read and Execute access. When you use templates in an ACL, place specific username(s) first and the most general usernames (templates) last.

The specific-to-general rule also applies when you *exclude* users via null access. For an example, here's the example given above: You want to retain owner access, and give all users Read and Execute access — except ANDY_B and any user whose username begins with $. You can type

```
) ACL EXAMPLE [!USERNAME],OWARE ANDY_B ,, $+,, +,RE ⤶
```

By specifying null for the usernames you want to exclude, you tell the system to exclude them before it acts on the +,RE template.

Any username template is like a group name. If you like the concept of group names, you can use username templates to achieve the group name effect. For example, for all users associated with project MARK_II, you could create profiles with usernames that ended in .MK2. With these usernames, you'd carefully avoid overlap with irrelevant usernames. Then, you'd configure default ACLs within the +.MK2 user directories to specify the kind of access you wanted for each person.

The number of user-access pairs you can specify is limited only by the number of characters involved. The maximum number of characters allowed is 255. You can assign a multiple-line ACL easily by writing it into a macro file with a text editor, ending each line with the & continuation character; then using the macro to assign the ACL to the desired file(s). For example, assume you create the macro ACL_MARK_II.CLI with the following ACL specification:

```
ACL/V (%-%) al_r,RE allan,WRE barnes,RE carnaby_a,RE erm,RE fort-,RE &
jamison,RE s.a.a.,RE prodigy,AE &
```

You could then assign the long ACL to file(s) quite easily; for example

```
) ACL_MARK_II PROG+ MESSAGES ⤶
```

This assigns the long ACL to all files whose names begin with PROG, and to file MESSAGES, in the working directory. To add OWARE access for the user who runs the macro, you'd insert the characters [!USERNAME],OWARE in the macro before the first username-access pair.

If you (or a user) want to change the ACL of all files in a directory and its subordinates, get into the directory and type an ACL command of the form

ACL/V # username-access *[username-access ...]*

For example,

) ACL/V # SALLY,OWARE OP,OWARE ↵

With Superuser on, you can bypass all ACLs. But so can any user who has Superuser privilege. The ideal arrangement allows people to work productively without Superuser.

After you've established the ACL standards that you want, the ACL mechanism works by itself. You don't need to change ACLs very often. But you do need to understand them.

User Groups (32–Bit CLI with AOS/VS II Only)

AOS/VS II offers user groups, but you will need to use the 32–bit CLI to use them. A user group is a set of users, associated by group name. Groups offer a simple way to handle access control for projects, since people can gain access to files by groupname, not username. Groups streamline file access control for group–oriented tasks like projects. Their main benefit is that they can eliminate the maintenance of long, intricate ACLs.

There are three main tasks involved in using groups:

- Set up groups by creating the group directory and group files that contain usernames;
- Set up ACLs of files so that groups work properly;
- Have users join the groups they want using the GROUPLIST command.

Details on these tasks follow.

1. Set up groups

   Decide on one or more group names; for example, MARK_II for a project named MARK_II. The group name, like a username, must be a valid filename of one to 15 characters.

   Make directory :GROUPS the working directory.

   Directory :GROUPS is not shipped with the operating system, so you must create it, using the command CREATE/DIR :GROUPS, before groups can be used. The :GROUPS directory need not be created again. As an ACL for :GROUPS, we suggest at least +,E. If you also use +,RE, any user will be able to list the group names. If this is acceptable, use an ACL of OP,OWARE +,RE. If you do not want users to list the group names, use an ACL of OP,OWARE +,E. Group users must then remember the specific group filename, since they will not be able to list files in :GROUP (unless they have Superuser privilege).

Using a text editor, create a text file named for the group and insert the usernames of all users you want in the group. Separate usernames by a CLI argument delimiter: space, tab, New Line, Carriage Return, comma, Form Feed, null, and/or a comment. The comment delimiters are braces — { and }.

For example, assume you want users JKM, CSTONE, ALAN_B, and MJR to be members of the MARK_II group. Use a text editor (SED, for example) to insert each name on its own line in file MARK_II. The MARK_II file contents look like this:

```
JKM      { Jan K. Mellon, project leader }
CSTONE   { Carol Stone, consultant }
ALAN_B   { Allen Bornstein, staff member }
MJR      { Martin Rameau, staff member }
```

Later on, insert other usernames or delete them as needed. The order of names in the file is unimportant.

As an ACL for a group file, we suggest creator,OWARE +:groupname,R — in this case, OP,OWARE +:MARK_II,R. Using groupnames in ACLs is explained next.

NOTE: Be careful with braces in comments; if you omit a leading or trailing brace, groups will not work properly, if at all.

2. Set up ACLs to allow group access

Both you and users must arrange for the ACLs of all files that the group will share to grant access to the group. The form to specify group access in an ACL is

username:groupname, access *[...]*

Template characters are allowed in both username and group name. For example, +:MARK_II,OWARE allows access to all members of the group. As usual with ACLs, the username lets you allow different kinds of access to different members of the group. For example,

JKM:MARK_II,WARE +:MARK_II,RE

gives user JKM WARE access to the file(s) and other members of the group RE access.

The order of username:group specifications in ACLs works the same way as the order of usernames. If you use template characters, you must order the username and group name specifications carefully to get the results you want. Generally, you will want to order them from specific to general. For example, take the following ACL:

JKM,OWARE CSTONE:MARK_II,WARE +:MARK_II,RE +,E

This ACL gives user JKM all access to the file; the OWARE specification, since it comes first, overrides any other specification (group or nongroup) for JKM. The ACL gives CSTONE WARE access to the file when she is a member of group MARK_II; it gives other members of group MARK_II RE access; and it gives all other users E access.

Regardless of a user's membership in a group, the user's default ACL remains as username,OWARE, or the value set with the DEFACL command. Files created by any user in the group directory structure will receive that user's default ACL. Depending on the default ACL, other group members may not be able to access such files. (Normally users will not want to change their default ACL at logon to grant access to group members, since this would allow group members to access files they created, even those in their user directories.)

So, for the group mechanism to work properly, group members may need to change their default ACLs when they join the group to provide access to other group members. Or someone with Superuser privilege can periodically change all ACLs within the group directory structure to allow access to all members.

3. Have users join groups as needed using the GROUPLIST command

   When a user logs on, his/her group list contains no group names. Users can learn the names of available groups from the system manager; if they have read access to :GROUPS, they can learn the names of all groups by typing FILESTATUS :GROUPS:+.

   To join a group, a user issues the GROUPLIST command with the group name as an argument. A group list can include up to eight group names. The GROUPLIST command switches let users insert or remove group names in their group lists, or kill their group list. If a user is not granted access to a group, the system will display the error message *User cannot be in group*; if the group does not exist, it will display *Group does not exist.*

   With the example above, when user JKM, CSTONE, ALAN_B, or MJR logs on, he or she can type GROUPLIST MARK_II to join the group. Any of these users will then have access to any file whose ACL contains (in the correct order) the specification username:MARK_II,access. Instead of typing the GROUPLIST command, they could insert it in a macro to execute when they wanted to join the group. In this macro, they could also change the default ACL so all files created would grant access to the group.

The group list, like the search list, is part of the CLI environment; you can specify a different group list on each level. A pseudomacro, !GROUPLIST, displays the current group list. The GROUPLIST command is further described in the manual *Using the CLI (AOS/VS and AOS/VS II).*

Benefits of Using Groups

As long as each username remains in the group file (above, GROUPS:MARK_II), the user can simply issue the GROUPLIST command to gain access to files that have username:group–name,access specified in their ACLs.

If any user leaves the MARK_II project, you can remove his or her access by deleting the username from the group file (instead of tediously changing all pertinent ACLs).

If a new person joins the project, you can simply add his or her name to the group file (again, instead of tediously changing the ACLs).

Group Access Example

This example builds on the group structure MARK_II outlined above. The group file MARK_II (in directory :GROUPS) specifies the users noted above. The contents of file :GROUPS:MARK_II are

```
JKM      { Jan K. Mellon, project leader }
CSTONE   { Carol Stone, consultant }
ALAN_B   { Allen Bornstein, staff member }
MJR      { Martin Rameau, staff member }
```

To begin, CSTONE logs on.

```
) DIR :UDD:MARK_II ↵                    (She makes MARK_II the working
                                         directory.)

) FILES/AS/S ↵                          (Tries to list files.)

Error: Read access is required          (Receives access error message.)

) GROUP MARK_II ↵                       (Joins group MARK_II.)

) FILES/AS/S ↵                          (Again tries to list files.)

. . .                                   (The CLI displays filenames.)

) SED PHASE2.OVERVIEW ↵                 (She runs the SED editor to create
                                         and edit a file.)
                                        (She edits the file and exits from SED.)

) ACL/V PHASE2.OVERVIEW↵                (She checks the ACL of her file.)
PHASE2.OVERVIEW CSTONE,OWARE            (This is her default ACL; it does not
                                         allow access to other group members.)

) ACL/V PHASE2.OVERVIEW [!USERNAME],OWARE & ↵

&)JKM:MARK_II,WARE +:MARK_II,RE +,E ↵
                                        (She adds access for group members:
                                         WARE for JJM and RE for the other
                                         members.)

) DEFACL PHASE2.OVERVIEW [!USERNAME],OWARE & ↵
&)JKM:MARK_II,WARE +:MARK_II,RE +,E ↵
                                        (She adds changes her default ACL
                                         to allow group access to other files she
                                         creates during this session. Actually,
                                         this command, like the GROUPLIST
                                         command, would probably be part of a
                                         macro she executed when she wanted
                                         to join the group.)
                                        (She continues to work in the group
                                         directory structure.)
) DIR/I MEMOS↵                          (She returns to her user directory
                                         structure.)
```

) GROUPLIST/K ↵	(She sets her group list to null — preventing access to group files that her username alone does not grant access.)
) DEFACL/V [!USERNAME],OWARE +,E ↵	(She restores her original default ACL.
GROUPLIST/K	This command and the command would probably be part of a macro she executed when she wanted to leave the group.)
. . .	(She continues to work in her user directory structure.)

Device and LDU ACLs

Devices — like disk and tape units, and terminals — are accessed through filenames in the peripherals directory (:PER). The default ACL of :PER is +, RE — but ACLs of files in it are more restrictive. For disk, diskette, and tape units the default ACL is OP,WARE, For terminals, it's PMGR,OWARE.

The default tape unit ACL (OP,WARE) allows users other than OP no access. To allow user access, many sites use the UP macro to change tape unit ACLs to +,WARE. Unfortunately, an ACL of +,WARE allows any user to read or write any tape that happens to be mounted on a unit — an obvious security risk.

One way to keep tape units secure (that is acceptable in a C2-level system) is to make users employ the MOUNT command, which asks the operator to mount a tape. After the operator mounts the tape and informs EXEC, EXEC changes the unit ACL to give the user exclusive access (username,WARE). The user retains exclusive access during the tape mount; then EXEC restores the old unit ACL.

You can restrict which users can use the MOUNT command by restricting access to the MOUNTQ. To do this, use the EXEC command ACCESS, with MOUNTQ and the username or template of the user(s) you want to give exclusive use of the MOUNTQ. For example, to give user Sally exclusive use of the mount queue, you would type CONTROL @EXEC ACCESS MOUNTQ SALLY OWARE and press NEW LINE. Sally, and no other user without Superuser privilege, could then submit mount requests. Any other user, except a superuser, would get a *Write access denied* error message after issuing the MOUNT command. You can restrict mount queue access to superusers by setting access to null (ACCESS with /K switch). Or to prevent any mount access, you can close the mount queue (EXEC CLOSE command).

Default disk and diskette unit ACLs — like tape unit ACLs — are OP,WARE. For diskette units — like tape units — many sites use the UP macro to change the unit ACL to +,WARE.

The ACL of a disk or diskette unit remains in force only while there is no initialized LDU in the unit. After a disk has been initialized into the file system, the ACL

assigned to the LDU with the Disk Formatter takes effect. (But for the system LDU, the root ACL is always +,E.) An initialized LDU is treated like any directory; its ACL can be changed by an owner of the parent directory. To initialize an LDU, a user needs Owner access to the LDU (set by the Formatter under AOS/VS and by Disk Jockey under AOS/VS II), Write access to the directory, and Execute access to the unit name(s) in directory :PER. See the *Installing* manual for your particular operating system for the details.

Before an LDU is initialized, it's vulnerable. Anyone at the system console or logged on with username OP can read from or write to it as a physical device (for example, by typing DUMP/V @DPJ10 MYFILE and pressing NEW LINE). To prevent this, have all on-line disks initialized as soon as possible after starting the operating system, or leave disk power off, or change disk unit name ACLs in :PER to null.

The operating system does not have an object reuse policy in regard to scratch tapes or diskettes. To prevent someone from reading a scratch tape or diskette, use a bulk eraser on the tape or diskette.

Any standard CLI process with username OP has owner access to many system files, allowing the process to do anything it wants with these files (ACLs of system files are described in Table 14-4). Also, EXEC will obey the commands of any process with username OP. EXEC commands can change ACLs on tape units, queues, and printers, or otherwise exert control. For these reasons, avoid leaving a process with username OP running unattended. On the system console, run a locked CLI.

The operating system lets magnetic media (tapes and diskettes) be handled in a secure fashion by privileged users *only* in a C2 configuration through the use of ACL and EXEC ACCESS commands. The default device ACL (OP, WARE) allows access only to the OP username. The OP username is uniquely assigned in a C2 system to a privileged user (usually the operator). The privileged user changes the default ACL (to privileged_username, WARE) via EXEC and mounts the media. The privileged user retains exclusive access during the media mount; then, EXEC restores the default device ACL (OP, WARE).

If processes use IPC files for I/O, you cannot protect the IPC files with ACLs. Instead, place the IPCs in a protected directory and make that directory the initial directory for the process. For example, you could create a directory for a user group MY_APP with an ACL of +:MY_APP,WARE. Start the process with the PROCESS command, adding /DIRECTORY=MY_APP.

Preventing Unauthorized Access to Windows

The operating system supports windowing on graphics (pixel-mapped) terminals on systems like the DS/7500 computer. On such systems, a user can create windows on a graphics terminal, but in order to do so, must first assign (get ownership of) the PMAP, an entry in :PER, for the device. Unauthorized access to PMAP entries represents a potential security risk because whoever has access to a PMAP entry can run programs (like a password-stealing program) in a related window. To prevent unauthorized access to windows, add to your system's UP macro a line that assigns PMAPs to PID 2. Place this line before the line that enables terminals.

Because of the security risk involved, windowing and graphics terminals are not permitted in a C2-level system.

ACLs of Operating System Files

The ACLs of files shipped with AOS/VS and AOS/VS II (and created by operating system utilities) are shown in Table 14-4. Do not change the ACLs of system files (except user ACLs) unless you have specific reasons for doing so. Changing them can imperil the security or operation of your system.

Table 14-4 ACLs of Operating System Files

File	Default ACL	Reason for Default and Comment
: (root directory)	+,E	Users need access to the root to log on to their own directories.(Execute access is required for access to any directory, with or without other access.)
:CLI files	+,RE	Users need to be able to read and execute the CLI to log on.
:CON0_LOG	OP,R	The operator needs to be able to read this file. If you want others to be able to read it, change its ACL to +,R in the system's UP.CLI macro.
:ERROR_LOG	null	Only a system manager with Superuser privilege should be able to run REPORT or a program that reads system error log records.
:HELP	OP,OWARE +,RE	The operator needs access without turning Superuser on. Users need to read Help files in the directory. This ACL is also used for all files in the directory. Possibly, you might restrict operator access and/or user access to some files.
LINK.PR (:UTIL)	+,RE	Programmers need Execute access to write applications. For Link, other program (.PR) files in :UTIL, and compilers and other directories, we recommend removing Read access from the ACL (for example, make the ACL +,E). Don't change the ACLs of overlay (.OL) files.
:LOCK_CLI.OL :LOCK_CLI.PR	OP,R (null)	A null ACL for LOCK_CLI prevents anyone but a superuser from reading it (and learning the password). The overlay file's ACL of OP,R lets the CLI program read it.
:PAGE	+,E	User processes must access this directory. Page files in the directory have null ACLs.
:PER	+,RE	User processes may need access to files in the peripherals directory (file ACLs follow).

(continued)

Table 14-4 ACLs of Operating System Files

File	Default ACL	Reason for Default and Comments
:PER:CONn	PMGR,OWARE	These are user terminal ACLs. The peripheral manager must own terminals to manage them. The peripheral manager assigns them to EXEC as users log on.
:PER:DPxn	OP,WARE	These are disk unit ACLs. The operator needs WARE access to initialize disk units. User processes *may* need access to diskette units; if so, you can change diskette unit ACLs, perhaps in the UP macro. Generally, only superusers should access hard disks, so you shouldn't expand hard disk unit ACLs.
:PER:MTxn	OP,WARE	These are tape unit ACLs. The operator needs to read and write to tapes. If users need tape access, a secure system would keep a restrictive ACL and require users to ask for labeled tape mounts, which lets EXEC supervise ACLs.
:PER:MRCDISKn	OP,WARE	These are Message–based Reliable Channel (MRC) ACLs. The operator needs WARE access to these devices. If users need access, a secure system would issue a restrictive ACL.
:PROC	+,E	User processes need to access this directory.
:QUEUE	+,AE	User processes need to add files (to be printed or run in batch) to this directory. The ACLs of files QUEUE_DESCRIPTORS and QUEUE_ENTRIES are +,R, to allow user processes to read QDISPLAY information. Users can read only the files that they have queued in :QUEUE.
:SWAP	+,E	User processes must access this directory. Swap files in the directory have null ACLs.
:SYSGEN	+,RE	Users may want to read AOS/VS system specification or AOS/VS II configuration files. While this directory is no great security risk, you may want to change its ACL to null — on the grounds that only a superuser will be generating a tailored system.

(continued)

Table 14-4 ACLs of Operating System Files

File	Default ACL	Reason for Default and Comments
:SYSLOG	null	Only a system manager with Superuser privilege should be able to run REPORT or a program that reads system log records.
:UDD	+,E	Users must be able to access their own directories.
:UPD	null	Only superusers like EXEC and PREDITOR should read or change profile files.
:UPDATE	+,RE	The operator and users must be able to read patch files in the directory and enter the directory. This ACL also applies to files in :UPDATE. You might consider giving :UPDATE a null ACL, since probably only a superuser will apply patches, and — possibly — a sophisticated user could learn points of weakness by reading patch files.
User directories	username,OWARE	Users should be able to do what they want in their directories. A user's files should not be accessible to other users. This username,OWARE ACL is the default ACL for all files created by the user in *any* directory where he/she creates files. You, or the user can change the default ACL with the command DEFACL. Default ACLs are described earlier in this chapter.
:UTIL (utilities)	+,RE	Users must be able to execute utility programs; the programs need to read overlay and data files. All files in :UTIL are shipped with an ACL of +,RE. If you want users to be able to add files to :UTIL, give :UTIL an ACL of +,ARE (or, more specifically, use username,ARE before the +,RE).

(concluded)

Aside from the recommendations above, you generally don't need to change the ACLs of files supplied by Data General and created by Data General programs.

Auditing with the System Log

Security doesn't maintain itself; it requires vigilance and review of system activity. The most important tool for review is the log file. This section outlines using the system log file and suggests procedures to follow. The section "Logging Operating System Events" in Chapter 11, goes into the details.

You can select logging with detail set to full or to minimal (default). Full detail logging will provide information about nearly everything that happened on the system. From a detailed log, you can create reports based on

- username
- file pathname
- failed logons

Logging without full detail produces a log that can tell you how much system time each user consumed, and who the privileged users are, but cannot tell you about file access or failed logon attempts.

SYSLOG also lets you use Superuser logging to log events, caused by a superuser, that would normally be logged only if you chose full–detail logging, and a protection feature, which prevents anyone from making changes to system logging while the system is up.

If you care about security, you will probably want to check for unwarranted file access or break-in attempts — so you will need full-detail logging at least occasionally. You may not always want to run the system log with detail set to full or with protection. Periodically, you should run it with detail set full to check for misuse of the system (like trespass on other users' or system directories). You should always run the log with detail full if you become suspicious about a user or file.

Logging Procedures

System logging (minimal or full detail) requires effort. Log files grow rapidly in a timesharing system, and with full-detail logging the system will panic if it runs out of log space. In any case, disk space is valuable. Here is an outline of the steps needed to use logging effectively.

1. Start logging. The SYSLOG command starts logging, and can rename the old log file. The system always writes user log information to file SYSLOG in the root directory. For full detail, add the /DETAIL=FULL switch to the SYSLOG command. SYSLOG is a privileged command; only the master CLI, PID 2, or a process with System Manager privilege can issue it. (Chapter 11 describes SYSLOG command syntax in detail.)

   Start logging as soon as possible in the UP macro and stop logging as late as possible in the DOWN macro. Do not start or stop SYSLOG while the multiuser environment is active.

   If you do start logging while the multiuser environment is up, SYSLOG cannot record information you need. If a process is running before SYSLOG starts, reports will identify the process by PID number only, not by username. And for a file open before logging starts, reports will include only the channel number and

PID, not pathname. Thus for both processes and files, reports might not make clear the identities of users or of objects on your system.

2. Monitor disk space use periodically. With full-detail logging, if the log file grows to consume all available disk space, the operating system will panic. Chapter 11 suggests using a separate directory or LDU for logging and explains a macro (supplied with the system) for monitoring disk space.

3. Generate a report from the log file as described in the next section.

4. At least weekly (better daily) dump the log file(s) and report(s) to backup media and delete them. Labeled media are best for this (described in Chapter 5). Log files are sensitive material, so store the log archives as securely as your other backup media.

   Old log files and reports can be very useful. Often, they can reveal when an intrusion started, and precisely what happened. Knowing these things gives you some picture of the damage — and, possibly some way to minimize this damage.

Creating and Interpreting Reports from Log Files

To get access to the information gathered by the system log files, you can write a program that examines the system log records (see an Appendix in the manual *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R Through ?Z*) but it is easier to use the REPORT program.

When you run the REPORT program, you can specify the log file(s), the output destination, and the type of information desired. (Do not run REPORT on the current SYSLOG file, because tbe operating system buffers SYSLOG events, and events in the buffer at the time REPORT runs will not be reported. Before running REPORT on the current SYSLOG file, close it and start a new one from PID 2 with the SYSLOG filename-for-old-syslog command line.)

To run REPORT, use the command form

[QBATCH] XEQ REPORT *[switches] [logfile-pathname] [...]*

where

QBATCH	Runs REPORT in batch. Depending on the size of the log file(s), REPORT may take some time (say 15 minutes for a 2-megabyte log file) to create the report on line. Thus, you might want to run REPORT in batch to keep your terminal free.
switches	Select options as described below.
logfile-pathname	Specifies the system (or error) log file from which the report will take its data. If you omit a *logfile-pathname*, REPORT will use all the log files in the working directory. (Log files are a special type of file, LOG.)

For a report on the current log file, you will need to turn Superuser on, since the ACL for this log file is null. (If the ACL of SYSLOG isn't null, make it null when you start logging, since the log file often contains sensitive information.)

When it generates a report from multiple log files, REPORT processes the files in chronological order of creation time — regardless of the order in which you specify them.

You can choose content for the report with switches. From the standpoint of security, the most useful reports include events by

- Username (Use switch /TRACE=username) The report lists all files the user accessed or tried to access, logons, processes created, pages printed, privileges turned on, file ACLs changed, and files deleted, with date and time. Since application programs run under specific usernames (like OP), a report by username can tell what files an application program accessed — a good way to check for Trojan horses.
- File pathname (Use switch /FILE=full-pathname) The report lists all users who accessed or tried to access the named file. The report gives username, date and time for each access, as well as printings, ACL changes, and deletion. This is useful as a check on sensitive files like :UPD, LOCK_CLI.PR,EXEC.PR, or your application program and data files.
- Failed logons (Use switch /FAILED_LOGONS) An increasing number of incomplete logon tries may indicate that someone's trying to break into your system.

The /AFTER and /BEFORE switches allow you to specify the time period to be reported on.

The switch /L alone sends the report to your generic @LIST file, while */L=pathname* sends the report to the file indicated by *pathname* (which could be a device, such as @LPT). If you do not use the /L switch the report goes to @OUTPUT, which, for interactive users, is your terminal screen.

A complete list of REPORT switches and the events they report appears in the section "Reporting Operating System Events (REPORT)" in Chapter 11.

If, in the report of an event, you see an error message, this indicates the user attempt failed. If there is no error message, the user attempt succeeded.

The choice of report content is very important. Generating a report can take a long time — 10 minutes, 20 minutes, even an hour. This practically eliminates spontaneous checking. But don't automatically include *everything* in your reports; aside from the resources and time required to generate large reports, too much material will overwhelm the person who reviews the reports.

If you care about security, you will probably want to check for unwarranted file access or break-in attempts — so you will need full-detail logging at least occasionally. You should always run the log with detail full when you become suspicious about a user or file.

If you see signs of trespass or break-in, follow them up carefully. You should have the power to react: discipline the offender(s) and tighten security after trespass, or tighten security (require password changes) after a break-in.

Protecting the System Site and Backup Media

Software and hardware protection features are effective only if the system hardware, and your backup media, are physically secure from user access. This section describes ways to protect user terminals, the system console, CPU, power source, disk and tape units, and backup media from unauthorized and possibly harmful access.

Users and Unattended Terminals

Anyone who is logged on a terminal and leaves it unattended risks all his or her files. Any passer-by can read or delete them. In a medium-security or closed shop, users should log off when they leave the site. It's a good idea to urge them to log off for lunch, or whenever their terminals will be unattended.

If a user has sensitive data or special privileges, he or she should *never* leave a terminal while logged on. It takes only a moment to copy a sensitive file into a common directory, where it can be studied later, at leisure. Also, a trespasser may be able to use a privileged process to learn passwords. Even an unsophisticated trespasser, if angry, can delete all the original user's files — which can take a lot of time to restore. (Files not backed up *cannot* be restored).

System Console and Locked CLI

If the system console is accessible to the user community, and runs a standard CLI, it's a serious security risk. An unlocked CLI on the system console has all privileges, or it can be terminated to expose a CLI with all privileges.

A locked CLI cannot be signed off or aborted while locked. While locked, it ignores commands that may threaten system security.

If you care about security, and your system console is accessible to users, you should run a locked CLI on it. Even if you have trusted operators, you may want to run a locked CLI as an additional security aid to limit the operator's powers.

The operation of the 16–bit LOCK_CLI is further described in Chapter 11, but how to set the password is only described here.

Changing the Password of LOCK_CLI (16–Bit CLI Only)

The program LOCK_CLI works only with the 16–bit CLI. For information about how to lock the 32–bit CLI, read the next section, "Locking the Master CLI (32–Bit CLI Only)." Like the standard CLI, LOCK_CLI consists of two files, LOCK_CLI.PR and LOCK_CLI.OL. It is in directory :UTIL.

The password supplied with LOCK_CLI is PASSWORD, but you should change this before running LOCK_CLI. To change the password of LOCK_CLI, you must edit file LOCK_CLI.PR with the FED disk file editor, put the new password in location PASSW, and end the password with a null (ASCII 0). Up to 31 filename characters, including spaces and control characters (but not CTRL-L) are allowed. They must all be uppercase.

The following example shows you how to change the original password (PASSWORD) to MAGIC.

Choose a display terminal. If this is a user terminal, log on with a profile that has Superuser privilege.

If this is the system console and it is running a locked CLI, unlock it by typing UNLOCK (and pressing NEW LINE); then PASSWORD (followed by NEW LINE). If you are in a normal CLI, proceed.

After FED starts, you can use its Help feature (type ESC H) if needed.

Press the ALPHA LOCK key on the terminal. Then type the following:

) SUPERUSER ON ⤵
Su) DIR : ⤵
Su) XEQ FED LOCK_CLI.PR ⤵

FED- Disk File Editor REV n

_ PASSW/ *050101* ⤵	Type PASSW/ (do not press NEW LINE) to see start of current password. The _ is the FED prompt, don't type it. FED displays the value on the same line following PASSW/. Press NEW LINE to continue.
BREAK/ESC T *00...01* _ 7 ⤵	Press the BREAK/ESC key and type T to open the display mode register. Then type 7 and press NEW LINE to make its value 7 for ASCII display. Press NEW LINE once more to continue.
_ PASSW/ *PA* _ "MA CR	Type PASSW/ again to see what the values are in ASCII. The first two characters are PA. Type a quotation mark ("); then type the first two characters of the new password. In our example, we typed MA; then press the Carriage Return (CR) key to open the next location.

PASSW+1/ SS _ "GI CR	Type a quotation mark; then the next two characters. In our example, we typed GI, then pressed Carriage Return to open the next location. (For a password with an even number of characters, type the last two characters, press Carriage Return; then type 0 (zero) to end the password, and press NEW LINE to close the location.)
PASSW+2/ WO 'C*400	C is the last letter in the example password. To specify one character, type an apostrophe (') rather than a quotation mark; then the character. Then, to put a null in the right byte, you must multiply the character by 400 as shown. The syntax is 'char*400 (As mentioned above, this isn't needed for a password with an even number of characters: you simply keep typing xx (where xx represents two characters) and pressing Carriage Return (until you've entered the whole thing; then type 0 and press NEW LINE to produce the terminating null in the next word.)
_ PASSW/ *MA* _ CR	When done with the password, go back and verify it. The first two characters are okay. Press Carriage Return to continue.
PASSW+1/ GI _ CR	The second two characters are okay.
PASSW+2/ C<0> ↵	The third two characters (last in the example) are C and null — okay. Press NEW LINE to close the display and mode register.
_ ESC Z	Press the ESC key and type Z to leave FED.
DONE!	
Su) XEQ LOCK_CLI ↵	Back to the CLI. Try the new password...
AOS/VS CLI REV n	
) SUPERUSER ON ↵	Try Superuser ...
)	LOCK_CLI ignores the command.

) UNLOCK ↵ MAGIC ↵	Type UNLOCK and the new password.
) SUPERUSER ON ↵	See if it's unlocked ...
Su) LOCK ↵	Yes; success. Lock it again.

)

If you cannot make the new password work, log on to a user terminal with a privileged profile, turn Superuser on, and use FED to check :UTIL:LOCK_CLI.PR, location PASSW, again; then try the UNLOCK again.

Note that file LOCK_CLI.PR should have a null ACL, but file LOCK_CLI.OL should have an ACL of OP,R so this CLI can read its own overlay file. The files are shipped with these ACLs, but you can assign them as follows:

Su) ACL/K LOCK_CLI.PR ↵
Su) ACL/V LOCK_CLI.PR ↵

LOCK_CLI.PR	(The CLI displays nothing, which means that the ACL is null.)

Su) ACL LOCK_CLI.OL OP,R ↵
Su) ACL/V LOCK_CLI.OL ↵

LOCK_CLI.OL OP,R	(The CLI displays an ACL of OP,R.)

Su)

Now, with the new password and correct ACLs, LOCK_CLI can safeguard the system console, user processes, and the system itself. To run it automatically, edit the UP.CLI macro to include it as described under LOCK_CLI in Chapter 11.

The people who operate the system don't absolutely *need* to know LOCK_CLI's password. If they don't know it, they won't be able to shut the system down normally from the system console — which might be what you want. (Anyone who has Superuser can get Superprocess, and terminate LOCK_CLI from a *user* terminal; so if the operator has a privileged profile, he or she can do this.)

In any case, the fewer people who know the password the better, because anyone who knows it can unlock, turn on Superuser and Superprocess, and use them.

LOCK_CLI retains the username OP, so it can issue commands to EXEC while locked. Thus, the operator can issue EXEC commands even while restricted to a locked CLI.

Locking the Master CLI (32–Bit CLI Only)

The locking technique for the 32–bit CLI is quite different from the one for the 16–bit CLI. The major differences between the two are

- the 32–bit CLI does not have a LOCK_CLI.PR file in the root (:) directory, and
- the 32–bit CLI uses a PASSWORD command, rather than the FED file editor, to specify a password

To specify a password for the 32–bit CLI, enter the 32–bit CLI, and type PASSWORD and then press NEW LINE. You will be prompted for a password of 6 – 15 alphanumeric characters. The 32–bit CLI will ask you to re–enter the password as a means of confirming the password. Now the 32–bit CLI has the password internally.

Now type PASSWORD/WRITE= and a filename, and press NEW LINE. The 32–bit CLI will write out the (encrypted) password to the specified file. You should then change the access to that file so that only the system manager and system operator(s) can read the file and the directory the file is in.

Now edit your UP.CLI macro to contain the command

PASSWORD/READ=filename

The 32–bit CLI will read the encrypted password every time UP.CLI runs.

Disabling the Break Sequence

Even with a locked CLI running, someone can type the break sequence, halt the CPU, and cause delays and lost data. Even worse, the person could run ESD, restart the operating system, and come up in the master CLI with all its powers.

You can prevent this by disabling the break sequence. This is useful only when the system console is separate from the CPU, because the effect of a break sequence can also be created by the use of CPU switches.

On ECLIPSE MV/10000 SX, MV/10000, MV/8000 II, MV/8000 C, and MV/4000 machines, use the CPU LOCK switch to disable the break sequence. On ECLIPSE MV/6000 machines, you should not disable the break sequence.

On an ECLIPSE MV/8000 machine, the only way to disable the break sequence is with the SCP command LOCK. You can do it at any time. If AOS/VS is running, enter the break sequence (CMD and BREAK, or BRK, or BREAK, depending on the system console); this gives control to the SCP-CLI. If AOS/VS is not running, simply type the SCP lock command. The SCP lock command is

SCP-CLI> FLAGS LOCK Y ↵
SCP-CLI>

The break sequence is now disabled; you can return to or boot AOS/VS as desired. The lock cannot be cleared until the SCP-CLI is active again and someone types

SCP-CLI> FLAGS LOCK N ↵
SCP-CLI>

A power outage to the CPU will also clear the lock. Pressing the CPU CONSOLE switch to RESET will give control to the SCP-CLI, regardless of the lock flag.

On ECLIPSE MV/4000 DC, MV/4000 SC, and Data General DS/4000-series systems, the only way to disable the break sequence is to type the SCP command FLAGS LOCK Y and press NEW LINE.

Don't disable the break sequence on these systems, however, unless it's critically important to deny access to the SCP. If the system hangs while the break sequence is disabled, you won't be able use break to enter the SCP and run ESD. To shut down, you'll need to turn power off, which means, if you are running AOS/VS, running FIXUP on all disks.

Computer and Power Source

Anyone who can touch the computer can unlock it, halt and reset it, run ESD, restart the operating system, and come up in the master CLI with all its powers. The system is only as secure as the people who have access to the CPU. In a closed or C2-level system, the computer should be locked away from the user community or have a trusted operator nearby.

The computer, and disk and tape units, are vulnerable to vandalism.

Cutting ac power to the system doesn't represent a security violation per se, but it may bring everything down — reducing productivity, increasing downtime and expenditures, and perhaps jeopardizing deadlines and projects. If you can, make your computer system power source secure (keep all power lines covered and out of plain sight).

Disk and Tape Units

Except for systems using the Message–Based Reliable Channel, cable lengths require disk and tape units to be fairly close to the computer. Thus, the same steps taken to protect the computer often serve to protect disk and tape units.

A user with access to disk units can bring down the operating system (by cutting power to the system disk or page/swap disk) or halt application-related work (by cutting power to other disks). If a disk unit has a removable pack, a user can remove the entire pack — perhaps to read it at leisure on a different system. Regardless of the disk type, a user with access to disks can harm the site and its productivity.

Tape units are always a risk — a user may be able to mount and read an accessible and sensitive tape.

Diagnostics and Maintenance Procedures

From time to time, you may run diagnostics (ADEX system tests) on your computer and peripherals. ADEX runs only on the system console, while user terminals are disabled. ADEX is designed to test computer hardware and detect errors without corrupting or changing user or system information. If ADEX is improperly used, however, it can erase information on disks and tapes. Therefore the person who runs it must be privileged, trustworthy, and knowledgeable.

Maintenance and preventive maintenance procedures (like changing printer paper, vacuuming, changing filters, and so on) require human access to the hardware. Anyone with this kind of access can start AOS/VS and steal or corrupt information. As with diagnostics, the people who perform maintenance must be trustworthy.

Backup and restoration procedures require the same kind of access as maintenance. The system is wide open to the person who does backups and restorations, because he/she requires (at least) Superuser privilege and, usually, access to the system console. As with maintenance, the people who perform backups and restorations must be privileged and trusted.

Storing Backup Media

Anyone with physical access to backup media has the opportunity to wreak havoc on your system. He or she needs only a tape or diskette unit (or compatible disk unit, for disk backups) to read the entire contents of your system: passwords, private databases — anything. Sophisticated users can use the DISPLAY program to learn tape- or diskette-set volume IDs and filenames, and they can use the LOAD command and pathname templates at leisure to get the information they want. Or, for revenge, they can destroy data on backup tapes (typing DUMP/V @MTxn:1 MYFILE to the first volume of a multivolume backup destroys the whole backup).

Access to backup media makes trespass and break-in relatively easy for experienced users who have access to devices (the devices can be on any Data General system). For a closed or C2-level shop, you should observe the same precautions as for the computer: lock up the backup media or keep it under the observation of a trusted operator.

For maximum safety, store backups away from the computer site, using a company that understands magnetic media storage. Keep backups near the computer for only a short period, say a week. Don't let users carry backup media (like tapes) away from the site. Lock up write-enable rings.

System Architecture — Hardware Protection Features

This section describes the protection features of ECLIPSE MV/Family computers that AOS/VS and AOS/VS II rely on to ensure security of code and data in memory.

MV/Family computers have a hierarchical memory protection mechanism, which protects the operating system, its Agent and peripheral manager, some server programs (INFOS II and DG/DBMS), and sensitive user data. This arrangement allows common routines to be used by all programs.

In a computer with the hierarchical protection mechanism, each user's address space includes a copy of the operating system. This lets user programs see the entire operating system as a subroutine — accessible via subroutine calls instead of specific system calls.

MV/Family computer memory is divided into eight units of 512 megabytes each, which are called segments. Between the segments are protection systems called *rings*. The lower-numbered segments are the most privileged; the higher numbered, least privileged. Ring 0 protects segment 0 and allows only privileged instructions to execute in segment 0. Ring 1 protects segment 1, ring 2 segment 2, and so on. The operating system kernel runs in ring 0 and its peripheral manager runs in ring 1. The Agent runs in ring 3, and INFOS II and DG/DBMS data management control programs run in ring 4.

User programs usually run in rings 6 and 7, where they have least chance to access operating system code and address space.

Whenever an operating sytem process refers to an address, the hardware checks the segment field of the address. The ring of the destination segment determines whether the source segment's reference is legal. This check occurs for any reference — a simple data reference or a transfer of control to the new segment.

A register called the segment base register (SBR) contains information about the validity of the referenced segment. If the reference is invalid, a protection fault occurs; the hardware blocks access to the segment, places the fault code 3 in AC3, and the operating system reports a protection fault to the process that attempted the access.

Accessing Other Segments for Data

A process can access data in another segment, depending on

- the process' location in memory;
- the protocol defined by the ring that protects the destination segment.

When the process makes the reference, the computer notes the source segment and destination segment, and determines if the source segment's reference is valid or not.

If not valid, a protection fault occurs, similar to the fault described above; the hardware blocks access, places the value 4 in AC3, and the operating system reports a protection fault.

The segment hierarchy (low to high, okay; high to low, fault) applies to access for data as well as other access. Table 14-5 shows which accesses are valid and which cause a fault.

Table 14-5 Valid and Invalid Segment Access

Process in Source Segment	Destination Segment							
	0	1	2	3	4	5	6	7
7	fault	fault	fault	fault	fault	fault	fault	valid
6	fault	fault	fault	fault	fault	fault	valid	valid
5	fault	fault	fault	fault	fault	valid	valid	valid
4	fault	fault	fault	fault	valid	valid	valid	valid
3	fault	fault	fault	valid	valid	valid	valid	valid
2	fault	fault	valid	valid	valid	valid	valid	valid
1	fault	valid	valid	valid	valid	valid	valid	valid
0	valid	valid	valid	valid	valid	valid	valid	valid

Ring Crossing

A ring can't be crossed accidentally, since the computer increments only the part of the program counter that doesn't involve the segment. The program must issue an LCALL instruction to attempt a crossing.

The hardware will allow a process to transfer to another segment only if all the following are true.

- The process must issue a subroutine call or return. No other calls will be accepted across a ring.
- The subroutine call must be inward (toward ring 0), and a return must be outward (toward ring 7). An outward call or inward return will cause a protection fault.
- A program must be running in the destination ring, and this program must have specified one or more *gates*, via a gate array. (A gate array is a series of entry points into the inner segment.) Thus, the inner-ring process must have *invited* access by the outer-ring one.

In the gate array, the inner-ring process specifies

- the number of gates;
- the range of segments that can use this gate; for example, the value 3 allows access by a process in ring 0, 1, 2, or 3. Processes in other rings will take a protection trap and access will fail;
- the address of the gate array.

When a process issues a ring-crossing call (LCALL), the computer checks its call against settings in the destination segment's gate array. If the crossing is allowed, it occurs. If the gate array doesn't allow it, a protection fault occurs in the *calling process*.

Protecting Against Hardware Trojan Horse Pointers

Suppose a process in segment 6 calls segment 2, and one of the arguments passed to segment 2 is a pointer to information in segment 2. When the process in segment 2 uses this pointer for reference, the computer will use segment 2's access privileges to determine the validity of the reference. These privileges may allow access to data *that would otherwise be restricted*. A pointer that attempts such access is called a Trojan horse pointer.

The operating system guards its kernel (in ring 0) and PMGR (in ring 3) against Trojan horse pointers, using hardware validate instructions. If you run application programs in inner rings, you might want to do the same (although processes are vulnerable to access only through gate arrays that they themselves define).

Indirection Protection

Whenever a process specifies an indirect address, the computer checks for a valid ring crossing. The first time each memory reference occurs, the computer treats the segment specified in the program counter as the source segment, and the segment specified in the intermediate address as the destination segment. The computer compares the fields; and if the access is invalid, the access is rejected and a protection fault occurs.

If the access is valid, the computer gets the new address specified by the intermediate address. Then, if another indirect address is specified, the computer treats the intermediate address as the source segment, and the segment specified in the new address as the destination segment (as described above).

As many as 15 levels of indirection are allowed. If a process tries more than 15 indirect address references, a protection fault occurs.

Page Protection

Page protection involves checks to see if an instruction in a process can legally access a page within the segment. There are two kinds of page protection: page-table entry validation and access validation.

Page-Table Entry (PTE) Validation

The validity bit (0) in the page table entry indicates whether the page is defined. If there's a reference to the page, and it isn't defined, this means it isn't in physical memory; a physical page fault occurs and the page is read from disk.

Access Validation

When a process tries to access a page that's in physical memory, the computer checks the page's access bits (2-4) for restrictions. For reads, it checks bit 2 (1 indicates a valid read, 0 an invalid read). For writes, it checks bit 3 (1 indicates a valid write, 0 an invalid write).

On transfer of control to the page, the computer checks bit 4 (execute). A 1 in bit 4 indicates a valid transfer, 0 an invalid transfer.

If the access bits forbid the type of access attempted, the access is blocked and a protection fault occurs.

Protection Fault Summary

ECLIPSE MV/Family machines have a number of protection fault codes, placed in AC1 when a protection fault occurs. These codes are summarized in Table 14-6. The operating system interprets the codes and returns an appropriate error message if the errors occur.

Table 14-6 MV/Family Hardware Protection Fault Codes

Fault Code (Octal)	Meaning and Description
0	Page read violation. The page-table entry denies Read access.
1	Page write violation. The page-table entry denies Write access.
2	Page execute (transfer) violation. The page-table entry denies Execute access.
3	Validity violation, ring reference or page-table entry.
4	Inward address violation. The process tried to access an inner ring.
5	Defer (indirect) violation, more than 15 levels of indirection.
6	Illegal gate. The maximum number of gates defined by the gate array has been reached.
7	Outward call. The process attempted a subroutine call through an outer ring.
10	Inward return. The process in the current ring tried to return, but the return address specified an inner ring.
11	Privileged instruction violation. A process not in segment 0 attempted a privileged instruction.
12	I/O protection violation. The process attempted an I/O instruction, but the segment base register of the current segment is set to prevent I/O instructions from executing.
14	Invalid micro interrupt return block.

Security Policies

This section is included to help you design a framework — a set of security policies for your organization — within which to implement the procedures described in the "Security Procedures" section.

Human Factors in Security

It's impossible to maintain security, in any shop, without the commitment and support of your key employees. This book can't tell you how to maintain morale — but, unless you do it, your site will be vulnerable from within.

Anger and a desire for revenge are often factors in security breaches. So is opportunity — people who might otherwise respect the security of your system can be tempted by the sight of a Superuser prompt or tapes that obviously hold a backup image of the system disk.

Each site needs a security program that motivates people to protect information and limits exposure to sensitive information.

Choosing Your Level of Security

At first, you may want to pursue maximum security — after all, why *not* provide the greatest amount of protection and privacy? Unfortunately, all security improvements cost something. The cost may be obvious (for example, hiring a system operator or buying an extra disk unit for logging) or hidden (for example, logging with full detail slows down general processing).

Users often find that very secure systems are the least friendly and hardest to learn. And, hiring full-time operators and providing physical security for your hardware is expensive.

Also, security controls alone won't give you a secure system. They can't prevent authorized users from stealing or destroying data. Your perceptions and judgments about the people you employ and create profiles for (your users) are at least as important as your implementation of security controls. This means a continuing commitment of time and energy to security.

On the other hand, the potential loss from lax security can be staggering. Users find it very unfriendly if your system is down for days while its file system is restored, or if your company suffers because competitors learn about its plans.

You must weigh user-friendliness and performance against data security, confidentiality, and integrity, and strike a good balance for your site. This will probably take some trial and error experience, and the arrangements you choose will need regular review.

Some major tradeoffs between closed and open systems follow:

Closed or Medium-Security System	Open System
High startup expense. The need to protect hardware forces you to hire system operators, build lockable areas and divert your attention from other things.	Low startup expense. Hardware doesn't need protection, and users can monitor themselves, avoiding the need for operators and your attention.
Inconvenience for users. Unprivileged profiles force users to seek help with file access from the operator or a privileged user.	Convenience for users. Privileged profiles allow users to handle file access problems by themselves (via Superuser or Change Username privilege).
Audit trail drawbacks. Logging consumes disk space. Checking log reports requires your time (or someone's time).	Easy logging or no logging. There's no need to check for security violations (although you might want to use log files for user accounting).
Lack of trust harms morale. Treating users and employees as if no one were trustworthy may cause resentment. Maintaining morale requires special effort, perhaps an entire program.	Implied trust of users is good for morale.
Sense of privacy and confidence can raise morale. User trust in the privacy of their files may increase productivity. For example, someone may do confidential work on the system instead of using a typewriter.	Wariness about open system harms morale. The possibility of trespass and lack of privacy may prevent users from using the system for confidential work. Or, data loss resulting from accidental or malicious trespass may disrupt work and schedules.
A secure system may lead to financial rewards. A specific standard of security (such as the C2-level) is required by some government and private organizations — without it, you don't qualify for contracts.	An open system may reduce opportunity for company growth. Your competitors may win contracts that require security standards.
Security standards limit the risk in computer operations. They can help you minimize chances of disastrous loss. Security standards may simply be a matter of common sense.	Without security standards, there's no practical limit to the loss you can suffer. Loss of data in essential files, or theft of confidential material, can be disastrous — to people, companies, and even nations.

Changing Security Levels

The easiest course is to define the security level you want before your system is built, and then to build the system and live with that level. But people's needs change, and things may evolve to the point where you need to change security levels.

To *increase* security, you'll need to edit user profiles, and you'll probably need to change ACLs. You'll need to run a locked CLI on the system console (if you don't run it already). You may need to run system logging with detail set to full. Some of the procedures are

- Tell your user community why you're tightening security.
- In user profiles, with PREDITOR, remove special privileges: Change Username, Superuser, Superprocess, and Access Devices.
- For each user who deals with sensitive information, run PREDITOR and choose password encryption. (If any of these users relies on a network to other Data General systems, encryption may interfere with network access. Password encryption won't prevent use of CEO Mail between systems, but encryption *will* prevent access by XODIAC's RMA and FTA agents to any system that doesn't run both AOS/VS Revision 7.00 or later and a revision of XODIAC that supports encryption. Read your XODIAC Release Notice to see if encryption is supported.

  If you determine that password encryption *will* interfere with network access, you may want to create a new, secure profile for the user and encrypt the password. Then, you can use the MOVE command to copy all files from the user's old directory to the new, secure one. From the new directory, change all ACLs to new-username,OWARE.

  Then, have the user delete all sensitive files from the old directory. In the future, have the user log on under the old username for network access, and under the *new* profile to work on sensitive files. (This may seem a lot of work, for both you and the user, but the increased security gained by password encryption and isolation from the network is worth the extra effort.)
- Perhaps, make ACLs more restrictive. For example, if the default ACLs have been [!USERNAME],OWARE +,RE, you might make them [!USERNAME],OWARE. You can change existing ACLs *en masse* with Superuser and the # template.

  Changing the default ACL is easy if you have a central logon macro: just put or edit the DEFACL command in the central macro. Using a central logon macro involves running a single macro for every user at logon (it's the initial IPC file for each user), and then having this macro execute a user-accessible macro in each user directory. Setting up a central logon macro is explained in the *Installing* manual for your particular operating system.

  If there is no central logon macro, put the DEFACL command in each user's logon macro.

  If multiple users need access to a file, you may want to use tailored ACLs that spell out usernames, instead of using templates. Doing this effectively may mean eliminating automatic ACL assignment (via DEFACL) for some users. These ACL steps can tighten security quite a bit, at the cost of some extra work. CLI macros (run periodically during the day or after hours) can help ease such operations.

- Safeguard the system console. Either run LOCK_CLI on it, or keep it under continuous supervision, or both.
- To monitor user activity, you may want to use system logging, with detail set to full. The reports from this can tell you which users and processes are trying to touch (and actually touching) specific files.
- You may need to check your applications programs more carefully than before, restrict use of network software, and eliminate certain amenities like guest profiles.
- You may want to design some kind of orientation program to raise user consciousness about, and appreciation of, security.

If you need to introduce a C2 security level, incorporate all of the measures described above; in addition, eliminate or avoid all of the following on your system:

- networking
- modems
- graphics terminals (and windowing)
- guest accounts
- software from bulletin boards

Software from bulletin boards, while not specifically prohibited by the C2 standard, represents a security risk. Avoid it.

See the section "C2-Level Systems" earlier in this chapter for more details.

Loosening security is (generally) easier than tightening it. You can't simply let things degenerate into chaos, though. Some of the steps involved in loosening security are

- Tell your user community why you're loosening security.
- Widen the scope of ACLs. For example, include Read and Execute access for all users (via +,RE) in each user's default ACL.
- Use logging with minimal detail or don't log at all.
- Unlock your hardware (if physically locked away from users) and reduce the time an operator is on duty.
- Run a standard CLI on the system console.
- Give special privileges to more users. This is last in the "loosening" list, since it's potentially the most harmful.

Detecting and Responding to Breaches of Security

Recognizing that your system is under attack — or that a breach has occurred — is as important as the steps taken to ensure security.

It's essential to understand the *types* of violation that can occur, and the types of people who are motivated to do it. Generally, there are three types of threats to security. They are

- probing
- irresponsibility
- break-in

Probing is often undertaken as a contest between a user and the system — an intellectual challenge. Many probers are motivated by curiosity and a desire to outwit system designers and security planners. They are often young and rarely malicious; they are not prototypical felons (although breaking into computer systems is a crime).

You can detect probing by checking reports from detailed system logs. Probing may involve logon attempts or attempted access to files.

Probing — Failed Logon Attempts

If the log shows many failed logon attempts on local terminals, the prober may be a current user trying to log on to a privileged account (look for users trying to log on to privileged accounts, like OP). Since the logon try was made locally, the person is probably (although not definitely) a user or a known person, since a stranger using a local terminal might be noticed and reported.

If the failed attempts occurred over a modem line or virtual terminal, the user may not be a current user but an unhappy ex-user or skilled break-in person who knows AOS/VS. The username recorded for the failed logon may offer some clues.

Probing — Attempted Access to Files

A large number of failed file accesses (to :UPD, :LOCK_CLI, or applications programs, for example) may indicate probing. The username is a matter of record, and you should talk with the user who has this username. To be tactful, say that you assume the account is being used by someone else, and have the user change passwords at once.

Then, if the failed accesses continue with the same username, this person is almost certainly the culprit; further disciplinary action is needed. If the failed accesses stop, either someone else was using the account and was foiled by the password change, or the user was doing the probing and has stopped. In any case, you win: you've stopped the probing.

Password stealing, described earlier in this chapter, is another kind of probing. It's usually discovered by users. The best defense against it is user awareness.

Irresponsibility

On most systems, there are users whose access rights allow them to take confidential or privileged information, or to corrupt data or files. If such users abuse their privileges (steal passwords or copy sensitive data files), they lack a sense of responsibility. The system's security controls aren't to blame.

If a user is irresponsible with his own account — by telling people his password or allowing them to see him type it — you can suggest that he be more careful.

You can minimize damage done by irresponsible users by restricting access as far as possible. Review user privileges and sensitive file ACLs periodically. As important as system control, though, is a user's behavior. Is he or she acting responsibly? Working competently? Looking for another job? If a user's status within your organization is about to decline (or end), you should think about reducing his/her privileges, or even deleting his/her account.

Break-In

A person who breaks into your system (penetrates the secure perimeter) is often highly skilled and motivated. There are several forms of break-in: by a successful prober; by a programmer who, through a Trojan horse, learns privileged passwords and confidential information; by an agent or covert operative who has planned the assault for a long time.

Whoever does it, a break-in is serious. Typically, the person doesn't leave obvious tracks — detection may be a matter of luck or occur from events that have nothing to do with your security arrangements. However, if you maintain security awareness, study the log file regularly, and watch for signs of break-in, you will improve your chances of detecting a break-in.

The person who probes or breaks in may be a student, middle manager, or agent of a rival organization or country. Some sample sites, people, roles, and goals follow.

College or high school	A student may break in, moved by curiosity, intellectual challenge or the appeal of free computer time. His or her resources include time and ingenuity.
Retail chain or industrial firm	A clerk may break in, moved by resentment. His or her resources include familiarity with the site and access to the system.
	A manager may break in, moved by desire for capital gain or by resentment. His or her resources include access to the system, its records, and — perhaps — privileges like Superuser to bypass all controls.
	An industrial spy may break in, moved by desire for capital gain or for trade secrets. His or her resources include computer expertise and money.
	An applications programmer may break in, moved by desire for capital gain or for trade secrets. His or her resources include computer expertise, intimate knowledge of the computer system, and access to applications code.

	An outsider may break in, moved by desire for trade secrets or by resentment. His or her resources include time and money.
Bank or Stock Exchange	A manager or officer may break in, moved by desire for capital gain, for inside information, or by resentment. His or her resources include familiarity with the system and money.
	An applications programmer may break in, moved by desire for capital gain, for inside information, or by resentment. His or her resources include computer expertise, knowledge of the system, and access to applications code.
Phone network, power grid, or military site	A terrorist and/or religious fanatic may break in, moved by desire to destroy order. His or her resources include dedication and money.
	An agent of an unfriendly nation may break in, moved by desire for strategic advantage and/or political gain. His or her resources include computer expertise, new technology, and money.

Students (hackers) are most likely to probe — and possibly break in. But they don't often do serious damage. Managers and bank officials often represent cases of irresponsibility — and possible probing. An outsider who attempts access to a phone network or military site represents the most serious threat — and will often have the resources to break through security safeguards.

Detecting Security Breaches

If your system is vulnerable to attack — perhaps under attack — warning signals usually come from several sources:

- users
- your own observations
- study of log reports.

User Observations

Users have the power of numbers. Often, there are many users. They can be very helpful (and occasionally frustrating, by reporting anomalies that have an obvious explanation). Users may report the following kinds of events:

- Files are missing or inaccessible.
- A logon message indicates that the user logged on when he or she did not. Or, at logoff, a user sees a *Other jobs, same username* message when he or she has not logged on to another terminal or submitted a batch job.
- A list of processes (displayed by the ? macro) shows a user logged on to a second terminal when he/she is not.
- Users find software and/or data files that they didn't write in their directories.
- Printed files with a user's username on the header are found, but the user didn't print them.
- Users of modem lines encounter busy signals more often.
- Off-hour users find the system unusually slow.

Any of these observations could indicate trouble, and you should explore it promptly, checking the complaint against reality. If the unusual situation really exists and doesn't result from user error, treat the situation as a real break-in attempt and take action to identify the perpetrator and his/her access method (compromised password, insider access, etc., perhaps using the log file).

If the user's report is a false alarm — perhaps a result of ignorance — don't castigate the user. User training is the province of system administrators and managers; the responsibility rests with you, not users. It's in both your interest and the user's to increase his level of competence.

In any case, don't insult the user. By doing so, you would lose his/her observation powers, reduce site morale, and perhaps create resentment that could lead to future break-ins.

Your Powers of Observation

As a person who cares about security, you should stay alert to conditions that may indicate future incursions. Here are some of these conditions:

- After you type ? and press NEW LINE, it displays a process with a username belonging to a user who could not be logged on (he or she is on vacation, or is logged on to a modem line and doesn't have a modem).
- The ? macro displays process names or batch streams unknown to you.
- System response changes dramatically and inexplicably from time to time.
- Media (tapes or diskettes) and/or listings are missing.
- You find strange software in directory :UTIL or your main applications directories.
- There's physical evidence of tampering — your papers have been rearranged, or material is missing from a locked closet.
- The system seems to be doing an unusual volume of processing in off hours; for example, it seems to be busy at 6:00 a.m. or 11 p.m.
- There are ACL changes on critical files, like LOCK_CLI.PR and directory :UPD.
- Your organization has obvious morale problems: lower earnings, layoffs, employee turnover, and/or reorganizations have produced a subdued or hostile atmosphere.

As with user-reported anomalies, each of these observations warrants exploration. You may find that there is a problem and take preventive or other action; or there may be a harmless cause.

Study of Log Reports

With log detail set to full, you can track every pertinent file access, every process creation, and every failed logon attempt. (Logging without detail set to full can yield a report of privileged users, and how much system time users consumed, but it cannot provide other security and access-related information.)

How to decide on the information to have reported is discussed earlier (details are in Chapter 11). System files like LOCK_CLI and :UPD, and Trojan horse candidates like text editors and applications programs, are good candidates. And you can get reports on ACL change attempts. (A list of operating system files and their default ACLs appears earlier in the chapter, in Table 14-4.)

When a Breach Occurs

When a breach occurs, or is attempted and fails, there is standard sequence to follow:

- Detecting the breach — you learn that someone is probing or has penetrated your security perimeter.
- Identification — you learn the identity of the offender(s).
- Defense — you adapt security arrangements to prevent further violation.
- Repair — you fix the damage as far as possible.

Detecting Failed Break-Ins

Acts that suggest failed break-ins include file browsing (going shopping in other users' directories), password guessing, use of password-stealing programs, and unusual activity on terminal lines.

Browsing is easy to identify from reports from a detailed log file. Browsing often results from natural curiosity, and users may browse without malicious intent. You should treat an innocent browser differently from a malicious one, although it may not be easy to distinguish the two. An easy rule of thumb is to treat first offenses as innocent, other offenses as a sign of — charitably — noncooperation. Browsing *is* theft, theft of information. The value of the information stolen, or vulnerable to theft, should determine the strength of your response.

Password guessing and the use of password-stealing programs are often detected by users. Password guessing produces many failed logon attempts, but not under the user's actual username.

Password-stealing programs may give clues to their authors — if you get a log file report on file access of the target user's username (described in "Password-Stealing Programs," earlier in this chapter).

When someone is trying to break in, via password guessing or stealing, detailed log reports can tell you at least the *username* tried. Sometimes this is an anonymous name, like OP. But it may suggest a real user on your system. If the attempt is occurring from local terminals, you might actually be able to catch the offender in action. If it's occurring over a modem line, identification is harder. You *can* get users to change passwords or phone numbers, or direct EXEC in the ENABLE command to stop (disable the line) after a given number of retries. The latter solution may interfere with legitimate users, however.

Sometimes, the only way to identify an outsider is simply to wait — taking no new steps to bar intrusion and lay a few traps — in hopes that somehow the perpetrator will yield his/her identity.

Detecting a Break-In

When you learn of a break-in, it feels like an assault — a violation or theft of something close to you. Coping with trauma and the threat will require user efforts, and your best efforts as a manager.

Detecting a break-in may be easy or not easy — some signs of break-ins are described in previous sections. When you suspect or confirm a break-in, consider the possibility that the perpetrator has been using your system for days or even weeks. Consider what discretionary information you've lost. Then, if you ran detailed logging, dig out the old reports and, maybe, load one or two old log files to generate reports that may tell you when the break-in first occurred.

Identifying the Offender

After a break-in has occurred, you must decide whether the offender is an insider or outsider. This distinction is important, since it determines your prevention measures. The first step is to accumulate and analyze the information you have:

- user observations,
- your observations,
- your knowledge of the attempt (How well does the intruder seem to know the system? How were break-in attempts made?),
- and your old log files and reports, which may give some history on the break-in attempts.

If all this information suggests someone, fine. You may want to discuss the situation with that person immediately.

Often, though, to make positive identification, you must take some risks. If you really need to identify the intruder, you will probably need to allow more break-ins, while you analyze the intruder's actions. If you're not running a detailed log, start doing so immediately. Check the log file for patterns and clues to motivation. If the break-in occurs under one or two usernames, you might be able to plant identity-revealing traps in the user logon macro.

Review your backup procedures — and consider more frequent backups — with the goal of minimizing any damage done.

Identifying an Outside Offender

If the intrusion is on a local terminal, increase surveillance.

If it's occurring over a modem line, it may be very difficult to trace the caller. Phone traces are expensive, time-consuming, and difficult, and often they don't work. Generally, identifying an intruder through the phone system is worth considering only when substantial cash or property loss is involved. In many cases, spending your energy on preventing recurrences is more cost-effective than traces through the phone system.

Defending Against Break-Ins

The best defense is awareness. Stay alert to changes in behavior and morale; and be willing to implement changes as user and project statuses change. Here are pointers in a few key areas:

- Have users change passwords — with frequency based on the sensitivity of their material.
- Double-check your system software. A sophisticated programmer may have encoded Trojan horses in applications software. Check creation dates of program files, and get a report from the detailed log file on *all file accesses made by those processes*.
- Double-check the UPD directory (this is where profiles and passwords are kept).
- Read the log reports.

Repair Afterwards

If you think (or know) a break-in has occurred, check for file or data damage and repair it if you can.

To do this, you must identify the files that were accessed; see if the files were changed or corrupted; and restore valid versions from backup media as needed.

If you've identified the offending username, the most thorough check is to get a report on all files accessed — from a few days before the time of the first break-in to the present. Some of these files you can verify by deleting and recreating them; for example, with a user's profile file (in :UPD), you can use PREDITOR to delete and recreate the username (*not the user directory!*).

For files whose contents you can't easily check (like application programs), load earlier files that were backed up before the break-in. Then use the FILCOM or SCOM program to check for differences — and explore the differences. This is easier with a source file (readable text differences are displayed) than a program file (disk location differences are displayed).

You can use source files for comparison if you're sure that the program file wasn't corrupted (via a disk editor), and that a deceptive object file wasn't used to build a Trojan horse into the program. One way to detect program file edits or object file creations is to check reports for access to the disk file editor (:UTIL:FED.PR) and compiler (compiler program files include the language name, and .PR suffix).

Be aware, though, that a sophisticated user can copy editor or compiler programs into his directory under different names, and run the copies (instead of the originals). To prevent this, don't allow Read access to program files — at most, use an ACL of +,E for .PR files.

Deleting a User Profile (Revoking an Account)

Deleting a user profile prevents a user from logging on. Deleting a user's profile is recommended (for security reasons and to reclaim disk space) when a user leaves your organization. Also, deletion is an option if you know or strongly suspect that the user has abused his or her account (perhaps by examining other users' files or running a password-stealing program).

Profile information, including username, password, and privileges, is stored in each user's profile file in directory :UPD. The user's *files* are stored in directory :UDD:username.

With PREDITOR, using the DELETE command, you can delete the profile (in :UPD). This prevents the user from logging on but leaves his/her personal files intact. PREDITOR then asks if you want to delete the user directory.

If you care about security, deleting the user's directory (which also deletes all its files) is an important step. You may want to back it up (using the DUMP command and :UDD:username:# template) before deleting it. Deleting the directory removes it from easy reach of superusers and reclaims disk space.

(If you store a user's directory not on the UDD disk but on another disk, like UDD1, with a link to UDD, you must delete the user's files manually. PREDITOR will not be able to do it for you.)

You can *temporarily* deny a user logon access by using PREDITOR to change his/her password (you need not know the old password to change it with PREDITOR). Another way to deny access temporarily is to change the ACL of directory :UDD:username to null. In the first case, the user will get an *Invalid Username/Password Pair* message on every logon attempt. In the second case, the user will get a *File access denied — Contact your System Manager* message.

In either case, the user will be unable to log on without consulting someone in authority. If the person in authority wants to reinstate the user, it's easy enough to tell the user the new password or change the ACL of :UDD:username back to username,OWARE (or whatever it was originally).

After deleting a profile, think about access that any current user may have given to the deleted username. This represents a potential security risk: if a profile is ever created with the original username, the person with this username will inherit the access privileges of the deleted user. For example, say users Al and JKM are involved in a project and Al gives JKM Read and Execute access to some files. JKM leaves the organization; the system manager deletes the JKM profile. Six months later, the organization wants to create a profile for someone whose initials are JKM. If the new account is created with username JKM, chances are that the new user will inherit Read and Execute access to Al's files (unless Al remembered to change the ACLs — and perhaps his default ACL, if it included JKM).

One way of preventing this inadvertent granting of access to new users is *never* to reuse a username. Another way is to check ACLs of all directories at the :UDD:username level after creating a new profile. Without at least Execute access to directory :UDD:username, a user can't access any file in :UDD:username.

Security Check List

Figure 14-1 summarizes the security points in this chapter. It's a list of one line questions, followed by check boxes, multiple-choice boxes, or fill-in blanks. The list is designed to serve both as a checklist and summary.

If you want to run a secure system, we suggest you fill each space (use pencil). Then, for perspective, or if you want to increase or decrease security at your site, you'll have a point of reference.

System Security Check List

1. My site is an

   Open shop___ Medium-Security Shop___ Closed Shop___ C2-Level Shop ___

2. My user community includes

   ___ User Profiles ____ CEO Users

   ___ Total Users (include guests and multiple users of one profile)

3. Users in the community fall into the following categories:
   (include users in more than one category as appropriate)

   CEO Users ____ Data Entry Clerks ____ Managers ____ Programmers ____

   Secretaries ____ Word Processing Typists ____

4. Hardware and software

4a. CPU model (e.g., MV/20000) _____

   It has ____ type _____ asynchronous controllers and ____ user terminals.

   (For example, "It has 4 type IAC-16 asynchronous ... and 50 user terminals."

   It has ____ model _____ disks on ____ controllers.

   (For example, "It has 6 model 6239 disks on 2 controllers.")

   It has ____ model _____ disks on ____ controllers.

   It has ____ model _____ disks on ____ controllers.

   It has ____ model _____ tapes. It has ____ model _____ tapes.

4b. Aside from AOS/VS, I run the following software:

   Ada___ BASIC___ C___ CEO___ COBOL___ DG/DBMS___ DG/SQL___

   FORTRAN 77___ INFOS II___ INTERNET___ PASCAL___ PL/I___ PRESENT___

   Sort/Merge___ SWAT___ TRENDVIEW___ XODIAC products X.25___ FTA___

   RMA___ VTA___ Others_____ _____ _____ _____ _____ _____ _____ _____

4c. My site's application programs are written in the following language(s):

   _____ _____ _____ _____ _____ _____ _____

5. Physical security. Are any of the following under supervision of a system operator (Y or N)?___

   First shift: CPU__ System console__ Disk units__ Diskette units__ Tape units__

   Second shift: CPU__ System console__ Disk units__ Diskette units__ Tape units__

   Third shift: CPU__ System console__ Disk units__ Diskette units__ Tape units__

DG-27157

Figure 14-1 System Security Check List (continued)

6. Physical security. Are any of the following locked away from the user community (Y or N)?

   CPU__ System console__ Disk units__ Diskette units__ Tape units__

6a. How do users access tapes?

   Via physical access to units___ Labeled tape mount requests through EXEC___

   Either physical access or labeled tape mount, depending on user ___

6b. How do users get their printed text from the printer(s).

   Via physical access to printer(s) ___ Distributed by operator(s) ___

7. Do you encrypt passwords in users' PREDITOR profiles?

8. How many system users have the following privileges?

   Change username__ Superuser__ Superprocess__ Access devices__ System Mgr__

8a. Write the username of each privileged person and check privilege(s).

   Username_____
   Change username__ Superuser__ Superprocess__ Access devices__ System Mgr__

   Username_____
   Change username__ Superuser__ Superprocess__ Access devices__ System Mgr__

   Username_____
   Change username__ Superuser__ Superprocess__ Access devices__ System Mgr__

   Username_____
   Change username__ Superuser__ Superprocess__ Access devices__ System Mgr__

   Username_____
   Change username__ Superuser__ Superprocess__ Access devices__ System Mgr__

8b. Do any of these privileged users have network or modem privileges?
   If so, write the username and check the privileges.

   Username_____ Network___ Modem___

   Username_____ Network___ Modem___

   Username_____ Network___ Modem___

   Username_____ Network___ Modem___

   Username_____ Network___ Modem___

DG-27157

Figure 14–1 System Security Check List (continued)

9. Have you changed any of the default ACLs supplied with or created by AOS/VS files? If so, specify the parent directory name and filename.

Parent directory information Name _____ ACL _____

Filename _____ Old ACL _____ New ACL _____

Reason for change _____

Parent directory information Name _____ ACL _____

Filename _____ Old ACL _____ New ACL _____

Reason for change _____

Filename _____ Old ACL _____ New ACL _____

Parent directory information Name _____ ACL _____

Reason for change _____

Parent directory information Name _____ ACL _____

Filename _____ Old ACL _____ New ACL _____

Reason for change _____

10. At your site, what would be the two most destructive security-related events and their (estimated) dollar values?

Event _____ $_____

Event _____ $_____

11. Does your site have a clearly defined security program (Y or N)? _____

11a. If there is a security program, do users know why there is one? ___

11b. If there is a security program, how do users feel about it (check one)?

Like it____ Accept it____ Dislike it___ Hate it____

11c. If there is a security program, name major reasons for it (past break-in, general prudence, sensitive data, contract requirements). Write as many as apply.

_____

_____

_____

12. Does your site run the system log?

Yes, with detail full___ Yes, without detail full___ No___

If no, skip to question 18.

DG-27157

Figure 14-1 System Security Check List (continued)

13. How often do you generate reports?

    Daily____ Two to five times weekly___ Weekly___ Biweekly___

    Monthly___ When it seems appropriate___ Never____

14. If your site generates reports, what kind do you generate?

    By username ____ By filename ____ Failed logons ___

    Other (specify switches) _____

14a. Why do you do it this way? _____

15. How often are the reports read?

    Daily____ Two to five times weekly___ Weekly___ Biweekly___

    Monthly___ When it seems appropriate___ Never____

16. How often are the log files dumped and deleted?

    Daily____ Two to five times weekly___ Weekly___ Biweekly___

    Monthly___ When it seems appropriate___

17. Where is the log directory (check one)?

    Root (:)____ Other CPD directory on master LDU ____

    Directory on a nonmaster LDU ___

17a. Approximate size by which the log file grows each day, blocks _____

18. Are you satisfied (generally) with the level of security maintained at your site (Y or N)? ____

Notes:

DG-27157

Figure 14-1 System Security Check List (concluded)

End of Chapter

Appendix A
Modem Support

The manuals *Installing, Starting, and Stopping AOS/VS* and *Installing, Starting, and Stopping AOS/VS II* explain how to generate an AOS/VS or AOS/VS II system wiith an asynchronous controller (like an IAC–8 or DRT) that supports modem control signals. This appendix explains

- The functions that the operating system provides on lines that support the modem control signals;
- The modem control signals, and how the operating system uses them;
- How the operating system establishes a modem connection;
- How the operating system forces or breaks a modem connection; and
- How you can use the CLI CHARACTERISTICS command to control or alter the connection with a modem.

Operating system modem support follows two industry standards: the EIA RS–232–C standard and the CCITT V.24 standard.

Modem Control Functions

The operating system supports the following functions on modem–control lines:

- Modem connection support:

  The operating system uses the modem control signals to determine when connections are established and broken.

- Dial–out support under a modem connection:

  The operating system allows applications to write commands to a dial–out modem, and read responses, before a connection is established.

- Half–duplex support under a direct or modem connection:

  In full–duplex mode, data travels between the host and remote device in both directions simultaneously. In half–duplex mode, data travels from host to remote device, or remote device to host, in one direction at a time. Modems usually use full–duplex mode. Half–duplex mode is useful for special situations, for example microwave hookups.

- Hardware output flow control under a direct or modem connection:

  Data General asynchronous controllers that support the modem control signals also have hardware support for output flow control.

- Hardware input flow control under a full duplex direct connection:

  The operating system provides an option to use the modem control signals for hardware input flow control. While this is a common practice in the industry, this use of the modem control signals is not compatible with the EIA RS–232–C and CCITT V.24 standards. For this reason, the operating system cannot support this option in combination with modem connection support or half–duplex support.

Some Common Modem Signals

In the sections that follow, we refer to a number of common modem hardware signals, identified by two– or three–letter abbreviations. Modem signals are said to be raised (asserted) or lowered (deasserted) relative to ground. Because different modems can use these signals in different ways, it may be useful to know how the operating system processes these signals. These are the common signals:

DTR — Data Terminal Ready.

The operating system raises this signal when the line is first opened, and lowers it on the last close. On a dial–in line, this signal tells an auto–answer modem to answer the phone when it rings.

With CCITT V.24 auto–answer modems, the operating system waits for the phone to ring before raising this signal. (See signal RI and CHAR/MRI, later in this appendix.)

RTS — Request To Send.

The operating system raises this signal when it wants to transmit data. In full–duplex mode, the operating system raises RTS for the life of the session. In half–duplex mode, the operating system raises RTS when it wants to transmit data, and drops RTS when it has transmitted all the data and the line turn–around time has expired. (The line turn–around gives the modem time to receive the last character and to transmit it to the remote receiver; see CHAR/TLT, later in this appendix.)

The operating system also supports using RTS in a nonstandard way to provide hardware input flow control. (See CHAR/HIFC, later in this appendix.)

CD — Carrier Detect.

The modem raises this signal when the quality of the carrier signal is good enough to transmit or receive data. The modem lowers this signal if the quality of the signal degrades or if the user hangs up.

The operating system monitors this signal to tell when a modem connection is established, and when it should break a connection because the quality of the line is poor.

CTS — Clear To Send.

The modem raises and lowers CTS to control the flow of output data.

This flow control is implemented in the controller hardware, not in the operating system. However, this hardware support is optional on some controllers, and the operating system does control the selection of this option. (See CHAR/HOFC, later in this appendix.)

DSR — Data Set Ready.

The modem raises this signal when it takes the phone off the hook. The modem should not raise DSR if DTR is absent, and should lower DSR if the host lowers DTR or the remote user hangs up.

The operating system monitors this signal to tell when a modem connection is initiated or broken.

RI — Ring.

The modem raises this signal when the phone is ringing.

With CCITT V.24 auto–answer modems, the operating system uses this signal to tell when to raise DTR. (See CHAR/MRI, later in this appendix.)

The Modem Connect Sequence

This section describes how the operating system establishes a connection with a modem and the characteristics that pertain as the connection is made.

When the operating system opens an asynchronous line for the first time, it raises the DTR and RTS signals. These signals tell the modem that the operating system is ready to receive a call. The characteristics affecting the modem connect sequence are as follows:

CHAR/HDPX — If the HDPX characteristic is set, the operating system does not raise RTS yet. In a half–duplex environment the operating system raises RTS when it starts to transmit and lowers it when the transmisson is complete. (Requires modem hardware support.)

CHAR/MRI — If the MRI characteristic is set, the operating system does not raise the DTR and RTS signals yet. It waits for the phone to ring.

The operating system will not allow any I/O to occur until a connection is established. If the program that opened the line issues a write request, the operating system holds (pends) the write until there is a connection. If a character comes in before a connection is established, the operating system ignores the character.

CHAR/MDUA — You can override this pending of I/O by requesting direct user access. This characteristic allows the application to access the modem directly (read and write characters) before a modem connection is established. Use this characteristic to run applications like DG/BLAST or DG/GATE that directly dial out on Hayes–compatible modems.

NOTE: Some controllers will not receive characters unless the modem has raised CD. This is a hardware restriction, and outside the control of the operating system. This problem can be solved with a special cable that ties CD high. Also many modems can be programmed to keep CD high.

When a user calls in (from a remote terminal and modem), the modem attached to the host answers the phone. The two modems synchronize, and the modem attached to the host raises the signals DSR and CD. When the operating system sees that these two signals are raised, it waits 2 seconds, and then acknowledges that a connection is established and unpends any pended write requests.

The modem may raise the two signals DSR and CD in either order. If the system sees the modem raise CD before it raises DSR, then the system ignores it. If the system sees the modem raise DSR before it raises CD, then the system allows the modem 40 seconds to raise CD. (If the modem doesn't raise CD within 40 seconds, the system forces a modem disconnect. See the section "Forcing a Modem Disconnect.") Finally, when the system sees both signals raised, it acknowledges the connection. If the system had been timing 40 seconds waiting for CD, it stops timing. The system waits 2 seconds to allow the modem to settle. Then it allows any outstanding write request to finish, and accepts input.

CHAR/AUTOBAUD — If the modem attached to the host computer supports multiple baud rates, it may match the calling modem's baud rate. This characteristic directs the operating system to try to match baud rates. Press NEW LINE three times (or Carriage Return (CR) three times), and the operating system will be able to detect the following baud rates: 300, 600, 1200, 1800, 2400, 4800, 9600, and 19200 bps.

CHAR/MDUA — In the dial–out case, the modem will raise DSR immediately, but may not raise CD until a modem connection is established. This may take a long time, so the operating system does not impose the 40 second time–out.

CHAR/TCC — You can change the 40 second time limit to other values through this characteristic.

CHAR/TDW — You can change the 2 second wait time to other values through this characteristic.

Once a connection is established, the line behaves exactly like a nonmodem line. It continues to act like a normal line until the connection is broken.

The Modem Disconnect Sequence

A modem disconnect sequence occurs when a communication line fails. This happens when either side hangs up or the communication equipment physically fails. The modem communicates this fact to the host by dropping either or both of the DSR and CD signals.

When the modem detects that the communication line is still open but the two modems aren't synchronized, it drops CD. This case is covered in the next section "Forcing a Modem Disconnect."

When the modem sees that the local terminal (in our case the host) has dropped DTR, or that the communication line is lost (e.g., the remote side hangs up), the modem drops DSR. When DSR drops, the system drops DTR and RTS, terminates the process logged on this terminal if there is one (more accurately, the process that was first passed the terminal on a ?PROC system call), and aborts any other current and future I/O requests on this line with the MODEM DISCONNECT error. When the line is completely closed and then opened the system will again raise DTR and RTS (see the section "The Modem Connect Sequence" earlier in this appendix).

Forcing a Modem Disconnect

There are three circumstances under which the operating system breaks a connection.

- When a user first calls in and the modem raises DSR, the system waits up to 40 seconds for the modem to raise CD. If the modem doesn't raise CD in that time, the system forces a modem disconnect.

  (The 40 second time limit is adjustable. See "CHAR/TCC" in the section "Modem Timing Functions," later.)

- When CD drops, the two modems are no longer communicating (for example, because the communications line is too noisy). The system gives the communication problem 5 seconds to clear up. If CD is not raised again in 5 seconds, the system forces a disconnect.

  (The 5 second time limit is adjustable. See "CHAR/TCD" in the section "Modem Timing Functions," later.)

- When an asynchronous line is closed, if this is a last close (that is, no other process has it open) then the operating system forces a modem disconnect.

The operating system forces a modem disconnect by lowering the DTR and RTS signals. A drop in DTR notifies the host modem that the host system is no longer ready to communicate over the modem line. The modem responds by hanging up the phone and lowering DSR. The operating system responds to the drop in DSR by processing a modem disconnect.

In other words, the operating system forces a modem disconnect by telling the modem to hang up the phone and report a modem disconnect condition. The operating system then handles the modem disconnect condition normally (see the section, "The Modem Disconnect Sequence").

Other Modem Disconnect Considerations

On a last close of the line, the operating system may not lower DTR and RTS immediately. The system will not lower DTR and RTS until all buffered output is written from the line's output ring buffer to the terminal. (Of course, if the user hangs up, DSR will drop, causing a disconnect condition. The system will process the disconnect immediately, and lower DTR and RTS without waiting.)

Once the system has transmitted all buffered output, it will lower DTR and RTS. At this point the modem should lower DSR. If DSR does not drop when DTR drops or when the user hangs up the phone on the other end, the operating system still thinks there is a connection outstanding. Therefore, all open requests on this line will receive the error *Modem disconnect in progress – cannot open* until DSR drops.

Once DSR drops, the system will wait another 10 seconds before raising DTR and RTS. This allows the modem time to get ready for the next connection. This functionality prevents some modem problems caused by raising DTR and RTS too soon after DSR drops.

(The 10 second time limit is adjustable. See "CHAR/THC" in the section "Modem Timing Functions," later.)

Using Terminal Characteristics to Control Modems

There are a number of terminal characteristics that play a role in modem support.

Modem Connection Support

Default: OFF

CHARACTERISTIC switches: /MOD and /CTD

Mnemonics: ?CMOD and ?CCTD

The modem characteristic, CHAR/MOD, instructs the operating system to provide modem functionality (described above) on this line. If this characteristic is on, the operating system will treat the line as a modem connection. It will raise or lower DTR to establish or break a connection. It will monitor DSR and CD to decide when the remote side has established or broken the connection, and treat the line accordingly. If the modem characteristic is off, the operating system will still raise DTR and RTS on the first open (see the section entitled "The Modem Connect Sequence" earlier in this appendix) but it will not perform any of the other modem–related functions previously described.

CHAR/MOD differs from CHAR/CTD only in the log–on privileges required. If CHAR/MOD is on, the operating system allows only users with the PREDITOR "Use modem" privilege to log on.

Half–Duplex Support

Default: OFF

CHARACTERISTIC switch: /HDPX

Mnemonic: ?CHDPX

NOTE: This characteristic and ?CHIFC (see the section "Hardware Input Flow Control") are mutually exclusive. This characteristic requires ?CSMCD (see the "Suppress Monitoring Carrier Detect" section.)

Modems usually operate in full duplex mode, but some modems and some situations require half–duplex mode.

If this characteristic is on, the operating system will only raise RTS while transmitting. If this characteristic is off, the operating system will always raise RTS.

Tie RTS to CD

Default: OFF

CHARACTERISTIC switch: /RTSCD

Mnemonic: ?CRTSCD

NOTE: This characteristic has no effect unless ?CHDPX is turned on.

If this characteristic is on, the operating system will raise RTS and transmit data only if CD is off. If CD is on, it will wait. If the characteristic is off, the operating system will not check CD prior to raising RTS.

Hardware Output Flow Control

Default: OFF

CHARACTERISTIC switch: /HOFC

Mnemonics: ?CHOFC or ?HRDFLC

If this characteristic is on, the operating system will enable the CTS option on the controller hardware. If it is off, the operating system will disable this hardware option. This option is not available on all controllers.

Hardware Input Flow Control

Default: OFF

CHARACTERISTIC switch: /HIFC

Mnemonic: ?CHIFC

This characteristic provides hardware input flow control using a de facto standard that does not meet the EIA RS–232 standard. This mechanism also requires special cables that connect the CTS pin on each connector to the RTS pin on the other connector. Either side can raise or lower RTS to effectively raise or lower CTS on the other side.

If this characteristic is on, the operating system will lower RTS when it wants the device to stop transmitting data. If it is off, the operating system will not lower RTS to control the flow of data received.

NOTE: This characteristic is mutually exclusive with ?CMOD and ?CHDPX.

Monitor Ring Indicator

Default: OFF

CHARACTERISTIC switch: /MRI

Mnemonic: ?CMRI

The monitor ring indicator characteristic, CHAR/MRI, instructs the operating system not to raise DTR and RTS until the phone rings.

Modems that meet the EIA RS–232–C standard recognize DTR as meaning "when the phone rings, answer it." Modems that meet the CCITT V.24 standard, however, interpret DTR as "take the phone off the hook right now." When CHAR/MRI is set, the operating system supports these modems by not raising DTR and RTS until the phone rings (that is, until RI, Ring Indicator is raised). If this characteristic is off, the operating system will raise DTR as soon as a process opens the line.

Direct User Access

Default: OFF

CHARACTERISTIC switch: /MDUA

Mnemonic: ?CMDUA

This characteristic provides direct access to the modem. If this characteristic is on, a user program can issue ?WRITE system calls to send data over the line before a connection is established. In this way, a user program can, for example, send dial–out commands to the modem.

Suppress Monitoring Carrier Detect

Default: OFF

CHARACTERISTIC switch: /SMCD

Mnemonic: ?CSMCD

This characteristic provides some half–duplex support. If this characteristic is on, the operating system will suppress monitoring CD for modem connection processing. If off, the operating system will monitor CD to determine if there is a need to force a disconnect.

Modem Timing Functions

The following five switches/offsets allow you to specify values, in milliseconds, for five timing–related modem functions. The times shown are the default values.

/TCC=40000 (?CMTCC)	CD on Connect Time–Out Value. Time to wait for CD on a modem connect.
/TCD=5000 (?CMTCD)	CD Time–Out Value. Time to wait for CD to be raised after it lowers.
/TDW=2000 (?CTDW)	Delay First Write Timer Value. Time to wait after connection before allowing I/O.
/THC=10000 (?CTHC)	Hold Modem Closed Timer Value. Time to wait after disconnect for a modem to settle.
/TLT=0 (?CTLT)	Line Turn–around Time. For half–duplex, the time to wait, after transmitting the last character, before lowering RTS. NOTE: CHAR/HDPX must be on to use this characteristic.

Potential Problems With Modems

Users may find that they can use the modem control signals in nonstandard ways (for example, set DSR high and only use CD), and still have correct modem operation. This may not work on all Data General configurations, and Data General does not guarantee that such nonstandard operations will continue to work from one operating system revision to another.

End of Appendix

Appendix B
XLPT Mapper Files

The EXEC printer process, XLPT, is the EXEC cooperative program that formats and prints files. The operating system has a limited translation feature that allows for the use of either the ASCII 7–bit or the ASCII 8–bit Data General International (DGI) character set. This translation facility, called mapping, allows you to print a representation of the DGI character set on 7–bit printers.

This appendix describes the internal format of XLPT mapper files for AOS/VS and AOS/VS II.

Mapper files tell XLPT how to print each character. See the description of the EXEC MAPPER command in Chapter 3 for how to enable a printer to use a mapper file. A mapper file contains a series of commands, one per line. All lines must end with a line terminator such as New Line or Form Feed, including the last line. Like the CLI, you can continue lines by inserting an ampersand (&) immediately prior to the line terminator.

Mapper File Statement Syntax

The syntax of a mapper file statement is presented below. Anything contained inside double–angle brackets (e.g. <<number>>), represents a template which can match a variety of actual strings. Because the grammar described here itself describes templates, the notation may become a little confusing. Therefore, actual mapper file statement characters are **highlighted**.

<<statement>> ::==
    <<null_statement>>
    <<comment_statement>>
    <<assume_statement>>
    <<print_statement>>

<<null_statement>> ::==

<<comment_statement>> ::==
    COMMENT (any text, up to the end–of–line)

<<assume_statement>> ::==
    ASSUME PRINTING <<name>>

<<name>> ::== **NONE** | **ASCII** | **DGI** | **ALL**

<<print_statement>> ::==
    PRINT <<template>>
    [**AS** <<replacement>> [<<over_clause>>]]

[**MOVES RIGHT** <<decimal number>>]

<<template>> must be a single character;
<<replacement>> must 32 characters or less;

<<over_clause>> ::==
    OVER <<replacement>>

<<template>> ::==
    "<<string>>**"**
    '<<string>>'

<<replacement>> ::==
    "<<string>>"
    '<<string>>'

<<string>> ::==
<<atom>>[<<string>>]

<<atom>> ::==
    <<character>>
    [**!ASCII** <<octal number>>]

<<character>> ::==
    (Any printable character. This is site–defined.)

<<name>> ::==
    <<letter>><<name_tail>>

<<name_tail>> ::==
    <<letter>><<name_tail>>
    <<digit>><<name_tail>>

<<number>> ::==
<<digit>><<number_tail>>
<<number_tail>>
    <<hexadecimal_digit>><<number_tail>>

<<letter>> ::==
    A | B | C | D | E | F | G | H | I | J | K | L | M |
    N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
    a | b | c | d | e | f | g | h | i | j | k | l | m |
    n | o | p | q | r | s | t | u | v | w | x | y | z |

<<hexadecimal_digit>> ::==
    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F

<<digit>> ::==
    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Mapper File Statements

The following sections explain each of the statements that are used in a mapper file.

The NULL and COMMENT Statements

The NULL statement lets you insert blank lines into mapper files to make them more readable. Similarly, the COMMENT statement records any remarks the creator of the mapper file has to make. This makes the mapper file easier to understand and maintain. COMMENT statements can include any text, up to the end–of–line; COMMENT statements cannot be continued to the next line. Both COMMENT and NULL statements can be included at any point in a mapper file.

The ASSUME Statement

The ASSUME statement makes global settings in a mapper file, and must come before any PRINT statements. XLPT defines only one ASSUME clause: ASSUME PRINTING <<name>>. To override XLPT's assumption that the initial character set in use is DGI, you might include in the mapper file the phrase

```
ASSUME PRINTING ASCII
```

No other ASSUME clause besides ASSUME PRINTING is defined by XLPT.

The PRINT Statement

The PRINT statement is the heart of the mapper file functionality. It combines multiple functions and options into a single statement. A PRINT <<template>> statement means that any character string matching <<template>> is to be conveyed to output. Options on the PRINT statement

- allow redefinition of how to print the PRINT statement;
- describe what effect the output has on the device state; and,
- describe what effect the output has on the translator's internal state.

The PRINT statement, in its simplest form, designates a single character for printing. It says that the character is to be printed as itself, and that it will cause the screen cursor or the printhead of the printer to move one character to the right. For example, the PRINT statement

```
PRINT "[!ASCII 33]"
```

prints the escape character (33 octal) as itself.

The AS and OVER Options

The AS and OVER options alter how a given character prints. For example, the statement

```
PRINT "[!ASCII 346]" AS "ae"
```

directs that "ae" be printed whenever a byte with the octal value 346 occurs. Similarly, the AS and OVER clauses can define two or more replacement strings for a template, which will be printed one over the other. For example,

```
PRINT "[!ASCII 360]" AS "n" OVER "~"
```

would print as "n" with the tilde (~) character directly over it. The number of OVER clauses permitted is implementation–specific; XLPT permits at most one OVER clause. Other mapper file users could permit more, or might preclude over–striking entirely by allowing no OVER clauses.

XLPT does not reparse or remap the replacement values for a character.

Thus, a replacement string can contain any character or character string, including some that could alter the internal state of some devices.

XLPT assumes that if the OVER clause is used that the two replacement strings are the same width when printed, but it cannot assure this, because one or both of the strings may contain nonprinting characters.

The MOVES Clause

The MOVES clause in a PRINT statement explicitly specifies what screen cursor or printer printhead motion will result when a character string (or its replacement, if specified) is printed. If a MOVES clause is not given in a PRINT statement, the motion is assumed to be the length of the character string or its replacement. By default all characters are assumed to be printable: a MOVES clause overrides this assumption.

A MOVES clause allows software to track a device's positioning. Therefore, an application that has no interest in tracking cursor position would not require MOVES clauses in its mapper files.

For example, if a DG–to–ANSI translation did not need to know what the screen cursor position was, no MOVES clauses would be needed in its mapper files.

But XLPT, which must keep track of both column and line numbers, needs information on what motion a printer will make when a particular character is printed.

If a nonprinting character or character string needs to be sent to a printer, it should be defined in the mapper file as

```
PRINT <<template>> MOVES RIGHT 0
```

so that right motion by the length of the character string is not assumed.

The units of horizontal motion are columns and the units of vertical motion are lines.

A single template or its replacement could contain several motion–causing characters, in which case, the MOVES clause should reflect the net motion of all the characters taken together. Note that the MOVES clause can describe only relative motion.

Macros

The "[" and "]" characters are reserved in mapper files for macro invocations. The syntax of the invocations is similar to CLI's macro and pseudomacro facility.

The only mapper macro supported by XLPT is !ASCII. It has the same function as the CLI !ASCII pseudomacro; that is, to define characters by value. The arguments are the same; at least one octal number. The statement below, for example, makes any null character in a printed file visible:

```
PRINT "[!ASCII 000]" AS "<000>"
```

Any embedded null value will be replaced with the five characters "<", "0", "0", "0" and ">".

How XLPT Uses Mapper Files

The system manager can create a mapper file using a text editor or any method desired. The mapper file should either be created in the :UTIL:FORMS directory or moved to that directory when it is to be used.

As Chapter 3 shows, the filename of a mapper file may be given on a CX START command, a CX MAPPER command or a QPRINT/MAPPER= command. The first two commands set the mapper file for all jobs except those with /MAPPER= specified. The last command sets the mapper file for use only with the job submitted.

Once it has read a mapper file and reduced it to an internal form, XLPT uses the mapper file when formatting nonbinary, nonpass–through jobs. XLPT takes one character at a time from the input file(s) and looks it up in the mapper table to determine how to process it. If the character is printable, its mapped value is printed.

Note that the mapped value is not reparsed and mapped in turn. Devices like printers and terminals are very context–sensitive, and interpretation of any character depends on characters that precede it in the byte stream. Mapped values must be treated carefully, and not, for instance, have a page break inserted in the middle of them by a formatter.

For example, <11> means horizontal tab to NEC spinwriters, but <33><11><n> is a horizontal positioning command. If <33><11><n> were a string replacement value, and XLPT was reparsing replacements, XLPT would attempt to expand the <11> in the middle of the replacement, when context has given it an entirely different meaning. To reparse a mapped value, therefore, would be to interpret each character in the mapped value out of context.

This restriction has implications which may not be obvious. If a mapped value contains a special character, the special character will not be acted upon by XLPT. This has numerous implications; line or page counts could be thrown off, for example. Character mapping is very powerful, but has the potential to cause havoc. All mapper files should be kept in the :UTIL:FORMS directory, where they can be modified only by privileged users.

Using Setup and Cleanup Strings

There are three different strings that you can use to set up and clean up mapper files. They are the Job, Copy, and Page strings. You specify these strings in a mapper file that is read in when a print job is submitted with the /MAPPER= switch.

Specify a setup or cleanup string for a mapper file as follows:

JOB SETUP STRING IS *[any characters you choose including [ascii]]*

The same syntax applies to the Copy and Page commands. To create a cleanup string, substitute the word cleanup for setup, as follows:

JOB CLEANUP STRING IS *[any characters you choose including [ascii]]*

Place the setup string before the header page on a print job and after XLPT VFU downloading if you are using a job setup string. Place the setup string at the top of a page after "top of form" has been reached only on the first page if you are using a Copy setup. For a Page setup, place the string at the top of each page after "top of form" has been reached.

The rules for cleanup strings are the same, with the exception that cleanup strings occur at the end of print jobs submitted with the /MAPPER= switch.

You cannot specify more than one of the same type of string in the same mapper file.

These cleanup and setup strings are not intended for placing information in the VFU hardware of printers, but they may be used to simulate VFU, and as cleanup files if you want.

Before a cleanup file can be used with the /MAPPER= switch, the user must create the cleanup file, place the cleanup file in the directory :UTIL:FORMS, and enable binary mode at the destination printer with the command format

CX BINARY *@device cleanupfile_name*

Thereafter, whenever a QPRINT/BINARY (or QPRINT/PASSTHRU) request is made, XLPT sends the cleanup file *after* (and not before) the print job.The only time XLPT sends the cleanup file *before* a binary print request, is the *first* binary print request issued after the user has created the cleanup file, placed it in the directory :UTIL:FORMS, and enabled the target printer with the CX BINARY command shown above. In this one instance, XLPT sends the cleanup file both before *and* after the binary print request. XLPT also sends a cleanup file after *each copy* of a multiple–copy binary print request. (Formerly, XLPT sent the cleanup file once, after all copies of a multiple–copy binary print request were completed.) Cleanup files are sent *only* with binary print requests. XLPT does not send the cleanup file for QPRINT requests issued without the /BINARY (or /PASSTHRU) switch.

How XLPT Handles Mapper File Errors

If there is any XLPT–detected error in a mapper file on a CX START command, XLPT reports the error to the operator log file and the cooperative process for that printer is not started. If there is an error in a mapper file set by a CX MAPPER command, the error is reported to the operator log file and the mapper file is ignored. If there is an error in the mapper file designated by a QPRINT/MAPPER=, the error will be printed on the job output and the job will be flushed from the queue.

Any error in reading a mapper file always returns the general error message

Invalid mapper file

Supplementing this is a secondary error message which gives details of the error. Examples of error texts are shown below. In most cases, the text also includes the line number that the error was detected on. Thus, an operator trying to use an erroneous mapper file might see the following text on the operator log file:

FROM PID 3 (EXEC): @LPB: Invalid mapper file
FROM PID 3 (EXEC): @LPB: Unknown keyword, line 5

XLPT can only check the syntax of mapper files. It cannot determine whether a mapper file is correct for a given printer. Table B–1, on the following two pages, lists the secondary messages (with an explanation of each) that XLPT generates if it finds syntax errors.

Table B–1 Mapper File Secondary Errors

Error Message	Explanation
Argument is too long	The current implementation of XLPT restricts character strings and replacement strings to 32 characters maximum. This limit applies to the actual string value, not as it is written in the mapper file, which may be longer due to embedded !ASCI macros.
ASSUME statement cannot appear after a PRINT statement.	The ASSUME PRINTING statement, if there is one, must precede all PRINT statements.
Characters must be replaced singly	XLPT restricts character mapping to one character at a time. The first character string in a PRINT statement must specify only one character.
Duplicate keyword	Only one each of the AS, OVER and MOVES clauses may appear in a single PRINT statement.
Duplicate PRINT statement for same character	A given character cannot be mapped or otherwise specified more than once per mapper specification.
Duplicate statement	There can be only one ASSUME PRINTING statement in a single mapper specification.
Error on mapper file	The error message text given when XLPT gets back an error from the operating system when opening or the mapper file. Also received when any error is encountered in an old mapper file.
Invalid syntax	A catch–all error for any syntax error on which XLPT cannot be more specific.
Missing bracket	Similar to missing quote. Arises when a closing quotation mark is seen before the closing bracket of a character string.
Missing quote	Either the opening or closing quote is missing from a character string.
Number is too large	The numeric arguments to the !ASCII macro must be between 0 and 377 (octal). Likewise, the argument to the PRINT MOVES RIGHT clause must be between 0 and 255 (decimal).

(continued)

Table B–1 Mapper File Secondary Errors

Error Message	Explanation
Special character may not be mapped	XLPT doesn't allow its special characters to be mapped or otherwise modified with a PRINT statement. These special characters are: 010 Horizontal tab 012 New Line 014 Form Feed 015 Carriage Return 022 DC2 (First character of a VFU command) 040 Space 377 XLPT embedded command
Unexpected character	Any character, including nonprinting characters, cause this error if it is not expected by XLPT.
Unexpected keyword	Returned when a known word is seen in a context where is it does not belong.
Unknown keyword	An unrecognized word was seen where a keyword was expected.
Width is inconsistent with replacement	This error arises only in cases such as: PRINT "[!ASCII 33]" AS "" MOVES RIGHT 5 A null string cannot cause a move right.

(concluded)

A Sample XLPT Mapper File

Figure B–1 is a sample XLPT mapper file. This mapper file converts the Data General International (DGI) character set, so that it can be printed on a standard ASCII 7–bit printer.

```
COMMENT  This is a mapper for printing the Data General
COMMENT  International (DGI) character set on 96–character
COMMENT  ASCII 7–bit printers.

ASSUME PRINTING DGI

PRINT '[!ASCII 246]' AS 'o' OVER 'x'
PRINT '[!ASCII 247]' AS 'c' OVER '/'
PRINT '[!ASCII 250]' AS 'L' OVER '–'

PRINT '[!ASCII 253] AS '!'
PRINT '[!ASCII 254] AS '?'

PRINT '[!ASCII 272]' AS ""
PRINT '[!ASCII 273]' AS "S" OVER "y"
PRINT '[!ASCII 274]' AS ""
PRINT '[!ASCII 275]' AS ""
PRINT '[!ASCII 276]' AS ""
PRINT '[!ASCII 277]' AS "^" OVER "|"

PRINT '[!ASCII 300]' AS "A" OVER ""
PRINT '[!ASCII 301]' AS "A" OVER ""
PRINT '[!ASCII 302]' AS "A" OVER "^"
PRINT '[!ASCII 303]' AS "A" OVER ""
PRINT '[!ASCII 304]' AS "A" OVER "~"
PRINT '[!ASCII 305]' AS "A"
PRINT '[!ASCII 306]' AS "AE"
PRINT '[!ASCII 307]' AS "C" OVER ","
PRINT '[!ASCII 310]' AS "E" OVER ""
PRINT '[!ASCII 311]' AS "E" OVER ""
PRINT '[!ASCII 312]' AS "E" OVER "^"
PRINT '[!ASCII 313]' AS "E" OVER ""
PRINT '[!ASCII 314]' AS "I" OVER ""
PRINT '[!ASCII 315]' AS "I" OVER ""
PRINT '[!ASCII 316]' AS "I" OVER "^"
PRINT '[!ASCII 317]' AS "I" OVER ""
PRINT '[!ASCII 320]' AS "N" OVER "~"
PRINT '[!ASCII 321]' AS "O" OVER ""
PRINT '[!ASCII 322]' AS "O" OVER ""
PRINT '[!ASCII 323]' AS "O" OVER "^"
```

Figure B–1 Sample XLPT Mapper File (continued)

```
PRINT '[!ASCII 324]' AS "O" OVER ""
PRINT '[!ASCII 325]' AS "O" OVER "~"
PRINT '[!ASCII 326]' AS "O" OVER "/"
PRINT '[!ASCII 327]' AS "OE"

|PRINT '[!ASCII 330]' AS "U" OVER ""
|PRINT '[!ASCII 331]' AS "U" OVER ""

PRINT '[!ASCII 332]' AS "U" OVER "^"
PRINT '[!ASCII 333]' AS "U" OVER ""

PRINT '[!ASCII 340]' AS "a" OVER ""
PRINT '[!ASCII 341]' AS "a" OVER ""
PRINT '[!ASCII 342]' AS "a" OVER "^"
PRINT '[!ASCII 343]' AS "a" OVER ""
PRINT '[!ASCII 344]' AS "a" OVER "~"
PRINT '[!ASCII 345]' AS "a"
PRINT '[!ASCII 346]' AS "ae"
PRINT '[!ASCII 347]' AS "c" OVER ","
PRINT '[!ASCII 350]' AS "e" OVER ""
PRINT '[!ASCII 351]' AS "e" OVER ""
PRINT '[!ASCII 352]' AS "e" OVER "^"
PRINT '[!ASCII 353]' AS "e" OVER ""
PRINT '[!ASCII 354]' AS "i" OVER ""
PRINT '[!ASCII 355]' AS "i" OVER ""
PRINT '[!ASCII 356]' AS "i" OVER "^"
PRINT '[!ASCII 357]' AS "i" OVER ""
PRINT '[!ASCII 360]' AS "n" OVER "~"
PRINT '[!ASCII 361]' AS "o" OVER ""
PRINT '[!ASCII 362]' AS "o" OVER ""
PRINT '[!ASCII 363]' AS "o" OVER "^"
PRINT '[!ASCII 364]' AS "o" OVER ""
PRINT '[!ASCII 365]' AS "o" OVER "~"
PRINT '[!ASCII 366]' AS "o" OVER "/"
PRINT '[!ASCII 367]' AS "oe"
PRINT '[!ASCII 370]' AS "u" OVER ""
PRINT '[!ASCII 371]' AS "u" OVER ""
PRINT '[!ASCII 372]' AS "u" OVER "^"
PRINT '[!ASCII 373]' AS "u" OVER ""
PRINT '[!ASCII 374]' AS "ss"
```

Figure B–1 Sample XLPT Mapper File (concluded)

End of Appendix

Index

Symbols

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Document Set

For Users

AOS/VS and AOS/VS II Glossary (069–000231)

For all users, this manual defines important terms used in AOS/VS and AOS/VS II manuals, both regular and preinstalled.

Learning to Use Your AOS/VS System (069–000031)

A primer for all users, this manual introduces AOS/VS (but the material applies to AOS/VS II) through interactive sessions with the CLI, the SED and SPEED text editors, programming languages, Assembler, and the Sort/Merge utility. *Using the CLI (AOS and AOS/VS)* is a good follow–up.

SED Text Editor User's Manual (AOS and AOS/VS) (093–000249)

For all users, this manual explains how to use SED, an easy–to–use screen–oriented text editor that lets you program function keys to make repetitive tasks easier. The *SED Text Editor* template (093–000361) accompanies this manual.

Using the AOS/VS System Management Interface (SMI) (069–000203)
Using the AOS/VS II System Management Interface (SMI) (069–000311)

For those working with preinstalled systems and those on regular systems who want an alternative to the CLI, the SMI is an easy–to–use, menu–driven program that helps with system management functions and some file maintenance tasks.

Using the CLI (AOS/VS and AOS/VS II) (093–000646)

For all users, this manual explains the AOS/VS and AOS/VS II file and directory structure and how to use the CLI, a command line interpreter, as the interface to the operating system. This manual explains how to use the CLI macro facility, and includes a dictionary of CLI commands and pseudomacros.

For System Managers and Operators

AOS/VS and AOS/VS II Error and Status Messages (093–000540)

For all users, but especially for system managers and operators of regular (as opposed to preinstalled) systems, this manual lists error and status messages, their source and meaning, and appropriate responses. This manual complements *Installing, Starting, and Stopping AOS/VS*; *Installing, Starting, and Stopping AOS/VS II*; and *Managing AOS/VS and AOS/VS II*.

AOS/VS and AOS/VS II Menu–Driven Utilities (093–000650)

A keyboard template to identify function keys. A number of system management programs—such as Disk Jockey, VSGEN, and the SMI—and the BROWSE utility use the function keys identified on this template.

Installing, Starting, and Stopping AOS/VS (093–000675)
Installing, Starting, and Stopping AOS/VS II (093–000539)

For system managers and operators of regular (as opposed to preinstalled) systems, these manuals explain the steps necessary to format disks, install a tailored operating system, create the multiuser environment, update the system or microcode, and routinely start up and shut down the system. *AOS/VS and AOS/VS II Error and Status Messages* and *Managing AOS/VS and AOS/VS II* are companions to these manuals.

Managing AOS/VS and AOS/VS II (093–000541)

For system managers and operators, this manual explains managing an AOS/VS or AOS/VS II system. Managing tasks include such topics as editing user profiles, managing the multiuser environment with the EXEC program, backing up and restoring files, using runtime tools, and so forth. This manual complements the "Installing" manuals, whether for regular or preinstalled systems.

Starting and Updating Preinstalled AOS/VS (069–000293)
Starting and Updating Preinstalled AOS/VS II (069–000294)

For those working with preinstalled (as opposed to regular) operating systems on all computers except ECLIPSE MV/3500™ DC and MV/5000 Series systems, these manuals explain how to start, update, and change certain system parameters. The manuals also help you interpret error messages and codes. Companion manuals are *Using the AOS/VS System Management Interface* and *Using the AOS/VS II System Management Interface.*

Starting and Updating Preinstalled AOS/VS on ECLIPSE® MV/3000 DC and ECLIPSE MV/5000™ DC Series Systems (069–000481)
Starting and Updating Preinstalled AOS/VS II on ECLIPSE® MV/3000 DC and ECLIPSE MV/5000™ DC Series Systems (069–000480)

For those working with preinstalled (as opposed to regular) operating systems on ECLIPSE® MV/3000 DC and ECLIPSE MV/5000™ DC Series computers, these manuals explain how to start, update, and change certain system parameters. The manuals also help you interpret error messages and codes. Companion manuals are *Using the AOS/VS System Management Interface* and *Using the AOS/VS II System Management Interface.*

If you have one of these computer systems, use the pertinent manual above; discard any other *Starting and Updating Preinstalled* manuals you receive.

Using the AOS/VS System Management Interface (SMI) (069–000203)
Using the AOS/VS II System Management Interface (SMI) (069–000311)

For those working with preinstalled systems and those on regular systems who want an alternative to the CLI, the SMI is an easy–to–use, menu–driven program that helps with system management functions and some file maintenance tasks.

For Programmers

AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?A through ?Q (093–000542)
AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary, ?R through ?Z (093–000543)

For system programmers and application programmers who use system calls, this two–volume manual provides detailed information about system calls, including their use, syntax, accumulator input and output values, parameter packets, and error codes. *AOS/VS System Concepts* is a companion manual.

AOS/VS Debugger and File Editor User's Manual (093–000246)

For assembly language programmers, this manual describes using the AOS/VS and AOS/VS II debugger for examining program files, and the file editor FED for examining and modifying locations in any kind of disk file, including program and text files. The *AOS/VS Debug/FED* template (093–000396) accompanies this manual.

AOS/VS Link and Library File Editor (LFE) User's Manual (093–000245)

For AOS/VS and AOS/VS II programmers, this manual describes the Link utility, which builds executable program files from object modules and library files, and which can also be used to create programs to run under the AOS, MP/AOS, RDOS, RTOS, or DG/UX™ operating systems. This manual also describes the Library File Editor utility, LFE, for creating, editing, and analyzing library files; and the utilities CONVERT and MKABS, for manipulating RDOS and RTOS files.

AOS/VS Macroassembler (MASM) Reference Manual (093–000242)

For assembly language programmers, this reference manual describes the use and operation of the MASM utility, which works under AOS/VS and AOS/VS II.

AOS/VS System Concepts (093–000335)

For system programmers and application programmers who write assembly–language subroutines, this manual explains basic AOS/VS system concepts, most of which apply to AOS/VS II as well. This manual complements both volumes of the *AOS/VS, AOS/VS II, and AOS/RT32 System Call Dictionary.*

SPEED Text Editor (AOS and AOS/VS) User's Manual (093–000197)

For programmers, this manual explains how to use SPEED, a powerful (but unforgiving) character–oriented text editor.

Other Related Documents

AOS/VS and AOS/VS II Performance Package User's Manual (093–000364)

For system managers, this manual explains how to use the AOS/VS and AOS/VS II Performance Package (Model 30718), a separate product that is useful for analyzing and perhaps improving the performance of AOS/VS and AOS/VS II systems.

Backing Up and Restoring Files With DUMP_3/LOAD_3 (093–000561)

For system managers, operators, and experienced users, this manual explains the DUMP_3/LOAD_3 product, separately available, which provides backup and enhanced restoration functions, including precise indexing of files on a backup tape set.

The CLARiiON™ Series 2000 Disk–Array Storage System with AOS/VS (093–002227)
The CLARiiON™ Series 2000 Disk–Array Storage System with AOS/VS II (093–002190)

For system managers, these manuals explain how to understand and/or configure and use a CLARiiON disk–array storage system with AOS/VS or AOS/VS II.

Configuring and Managing DG/FTAM (093–000817)

For system managers, this manual explains how to configure, start, and manage DG/FTAM, the Data General implementation of ISO 8571—the File Transfer, Access and Management (FTAM) standard of the International Organization of Standardization.

Configuring and Managing the High-Availability Disk-Array/MV (H.A.D.A./MV) Subsystem (014–002160)

For system managers, this manual explains how to understand and/or configure and use a H.A.D.A./MV disk–array storage system with AOS/VS II.

Configuring Your Network with XTS (093–000689)

For network administrators, managers, or operators responsible for designing, configuring, or maintaining a network management system, this manual describes how to manage and operate Data General's XODIAC™ Transport Service (XTS and XTS II) under AOS/VS and AOS/VS II.

Installing and Administering DG TCP/IP (093–701027)

For network managers and operators, this manual explains how to install and manage a TCP/IP network under AOS/VS.

*Managing and Operating the Data General/PC*Integration Network Under XTS II* (093–000814)

For network administrators, managers, and operators, this manual describes the tasks for managing and operating the Data General/PC*Integration network on an ECLIPSE® MV/Family computer.

Managing and Operating the XODIAC™ Network Management System (093–000260)

For network managers and operators, this manual describes how to install and manage the Data General proprietary network software.

Managing AOS/VS II ONC™ /NFS® Services (093–000667)

For network managers and operators, this manual explains how to install and manage ONC Network File server software under AOS/VS II.

Managing AOS/VS II TCP/IP (093–000704)

For network managers and operators, this manual explains how to install and manage a TCP/IP network under AOS/VS II.

Programming with the AOS/VS II TCP/IP Sockets Library (093–000820)

For experienced network programmers, this manual provides information necessary to write applications that use socket library calls to access the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

Programming with the Remote Procedure Call (RPC) on AOS/VS II (093–000770)

For experienced network programmers, this manual provides information necessary to write the Remote Procedure Call for the AOS/VS II UDP/IP and TCP/IP networks.

Programming with the Transport Layer Interface (TLI) on AOS/VS II (093–000826)

For experienced network programmers, this manual provides information necessary to write applications that use the set of calls that access the Transport Layer Interface (TLI), Data General's implementation of the AT&T Transport Layer Interface. This implementation of the TLI is provided by DG/OSI Transport Service (DG/OTS).

Using CLASP (Class Assignment and Scheduling Package) (093–000422)

For system managers, this manual explains how to use the AOS/VS and AOS/VS II Class Assignment and Scheduling Package (Model 31134), a separate product that is useful for tailoring process scheduling to the needs of a specific site.

Using the MV Data Center Manager (093–000769)

For system managers, this manual explains how to use the MV Data Center Manager software, a separate product that manages multiple ECLIPSE MV/Family computers from an AViiON workstation.

TO ORDER

1. An order for documentation can be placed with the Technical Information and Publications Services (TIPS) in three ways:
   a. Mail Order – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form. Send your order form with payment to:

      Data General Corporation
      ATTN: Educational Services/TIPS A131
      4400 Computer Drive
      Westboro, MA 01581–9973

   b. Telephone – Call TIPS at 1–800–343–8842, Option #4, for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Training representatives are available from 8:30 AM to 5:00 PM EST.
   c. Fax – You can fax your order to (508) 898–4244 for quick, convenient service.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a. Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.
   b. Check or Money Order – Make payable to Data General Corporation.
   c. Credit Card – A minimum order of $20 is required for MasterCard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

	Shipping & Handling Charge	
Total Quantity	**UPS**	**Airborne (Overnight)**
1–5 Items	$5.00	$15.00
6–10 Items	$10.00	$25.00
11–40 Items	$15.00	$35.00
41–100 Items	$50.00	$70.00
Over 100 Items	$100.00	Call for price

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order. This schedule does not apply to Maintenance Service Guides and Maintenance Documentation Packages (MDPs).

Order Amount	Discount
$500–$999.99	10%
Over $1000.00	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at 1–800–343–8842, Option #4, to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing. Canadian customers should call 1–800–565–3583.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS A131
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	**SHIP TO:** (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME _____	COMPANY NAME _____
ATTN: _____	ATTN: _____
ADDRESS _____	ADDRESS (NO PO BOXES) _____
CITY _____	CITY _____
STATE _____ ZIP _____	STATE _____ ZIP _____

Priority Code _____ (See label on back of catalog)

Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

Check Shipping Method ☐ UPS Airborne (Overnight)

# of ITEMS	ADD	ADD
1–5 Items	$5.00	$15.00
6–10 Items	$10.00	$25.00
11–40 Items	$15.00	$35.00
41–100 Items	$50.00	$70.00
100+ Items	$100.00	*Call for Charges

These rates apply to 48 contiguous states.

B VOLUME DISCOUNTS*

Order Amount	Save
$500.00–$999.99	10%
Over $1000.00	20%

Tax Exempt # or Sales Tax (if applicable)
_____

ORDER TOTAL	
Less Discount See B	–
SUBTOTAL	
Your local** sales tax	+
Shipping and handling – See A	+
TOTAL – See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is _____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number _____ Expiration Date _____

Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Does not apply to maintenance documentation.

** Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 1–800–343–8842.

Form 702
Rev. 9/94

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE

TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub–licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision–locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Managing
AOS/VS and
AOS/VS II

093–000541–03

Cut here and insert in binder spine pocket