UNISYS

A Series System Administration Guide

Release 3.9.0

September 1991

U S America

Priced Item

8600 0437-000

UNISYS

Product Information Announcement

o New Release o Revision • Update o New Mail Code

Title

A Series System Administration Guide

This update for the *A Series System Administration Guide* is relative to the Mark 4.0.0 release of the A Series software.

This update includes new and changed information describing the Mark 4.0.0 system software.

The major changes described in this update are the following:

- New user billing capabilities
- Additional LOGGER capabilities
- Description of the LOCKEDFILE file attribute

Various technical and editorial changes have been made to improve the quality and usability of the document. Changes are indicated by vertical bars in the margins of the replacement pages.

Remove	Insert
iii through iv	iii through iv
vii through viii	vii through viii
xi through xx	xi through xx
1–9 through 1–18	1–9 through 1–18
3–3 through 3–4	3–3 through 3–4B
4–1 through 4–2	4–1 through 4–2
5–7 through 5–8	5–7 through 5–8B
5–17 through 5–18	5–17 through 5–18
5–21 through 5–22	5–21 through 5–22
6–1 through 6–2	6–1 through 6–2B
6–7 through 6–8	6–7 through 6–8
6–15 through 6–18	6–15 through 6–20
8–1 through 8–14	8–1 through 8–2
9–3 through 9–6	9–3 through 9–6
9–11 through 9–14	9–11 through 9–14
10–7 through 10–8	10–7 through 10–8
11–5 through 11–8	11–5 through 11–8
12–1 through 12–4	12–1 through 12–4
12–9 through 12–10	12–9 through 12–10
12–13 through 12–24	12–13 through 12–26
Glossary–7 through 8	Glossary–7 through 8
Glossary–31 through 32	Glossary–31 through 32

continued

Announcement only:

Announcement and attachments:
AS145

System: A Series
Release: Mark 4.0.0 July 1992

Part number: 8600 0437–010

Printed on recycled paper

To order additional copies of this document

- United States customers call Unisys Direct at 1-800-448-1424
- All other customers contact your Unisys Subsidiary Librarian
- Unisys personnel use the Electronic Literature Ordering (ELO) system

UNISYS

A Series System Administration Guide

Release 3.9.0 September 1991

U S America
Priced Item 8600 0437-000

The names, places, and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual, living or otherwise, or that of any group or association is purely coincidental and unintentional.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication may be forwarded using the Product Information card at the back of the manual, or may be addressed directly to Unisys, Product Information, 25725 Jeronimo Road, Mission Viejo, CA 92691.

Page Status

Page	Issue
iii through iv	–010
v through vi	–000
vii through viii	–010
ix	–000
x	Blank
xi through xix	–010
xx	Blank
1–1 through 1–8	–000
1–9 through 1–18	–010
2–1 through 2–8	–000
3–1 through 3–2	–000
3–3 through 3–4	–010
3–4A	–010
3–4B	Blank
3–5 through 3–10	–000
4–1 through 4–2	–010
4–3 through 4–9	–000
4–10	Blank
5–1 through 5–6	–000
5–7 through 5–8	–010
5–8A	–010
5–8B	Blank
5–9 through 5–16	–000
5–17 through 5–18	–010
5–19 through 5–20	–000
5–21 through 5–22	–010
5–23 through 5–37	–000
5–38	Blank
6–1 through 6–2	–010
6–2A	–010
6–2B	Blank
6–3 through 6–6	–000
6–7 through 6–8	–010
6–9 through 6–14	–000
6–15 through 6–19	–010
6–20	Blank
7–1 through 7–15	–000
7–16	Blank
8–1 through 8–2	–010
9–1 through 9–2	–000
9–3 through 9–6	–010
9–7 through 9–10	–000

continued

Page Status

continued

Page	Issue
9–11 through 9–14	–010
10–1 through 10–6	–000
10–7 through 10–8	–010
10–9 through 10–21	–000
10–22	Blank
11–1 through 11–4	–000
11–5 through 11–8	–010
11–9	–000
11–10	Blank
12–1 through 12–4	–010
12–5 through 12–8	–000
12–9 through 12–10	–010
12–11 through 12–12	–000
12–13 through 12–25	–010
12–26	Blank
13–1 through 13–5	–000
13–6	Blank
Glossary–1 through 6	–000
Glossary–7 through 8	–010
Glossary–9 through 30	–000
Glossary–31 through 32	–010
Bibliography–1 through 3	–000
Bibliography–4	Blank
Index–1 through 16	–000

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678–*xyz*) indicates the document level. The first digit of the suffix (*x*) designates a revision level; the second digit (*y*) designates an update level. For example, the first release of a document has a suffix of –000. A suffix of –130 designates the third update to revision 1. The third digit (*z*) is used to indicate an errata for a particular level and is not reflected in the page status summary.

About This Guide

Purpose

This guide is written to help administrators of A Series systems make basic decisions about how to operate their systems and their sites. After reading both this guide and the *A Series System Operations Guide*, system administrators should have a basic understanding of A Series administration and operation tasks.

Scope

This guide describes your tasks as a system administrator and provides information to help you perform those tasks. The guide distinguishes between required, recommended, and optional tasks.

In some cases, you are referred to another manual for information about a particular administrative function. This guide distinguishes among the following types of administrators:

- System administrator
- Security administrator
- Database administrator
- Network administrator
- Mail administrator

Although some or all of these functions are likely to be handled by one person at your site, information specific to the administration of your security system, your database management system, your network, and your electronic mail system can be found in the libraries for those products.

Audience

This guide is written for any person who is responsible for making administrative decisions at an A Series system site.

Prerequisites

You should be familiar with A Series hardware and system software. If you are not, read the *A Series Systems Functional Overview* before you read this guide.

Ideally, you should read this guide before your system is delivered and installed. If doing so is not practicable, read this guide as soon as possible after using the software installation guide for your system. Have the *System Operations Guide* available for reference while you read this guide.

How to Use This Guide

Read Section 1, "Identifying Initial Administrative Strategies," to familiarize yourself with the different types of administrative tasks. Every site must perform the tasks identified in the "Required" category. Whether or not the tasks identified in the "Recommended" and "Optional" categories are performed depends upon the needs of the individual site.

Read the remaining sections on specific topics to identify the individual tasks. Those sections also give information or point to other manuals that give information about how you might want to perform the task for your site.

Organization

This guide contains the following 13 sections. In addition, a glossary, a bibliography, and an index appear at the end of this guide.

Section 1. Identifying Initial Administrative Strategies

This section provides an overview of the administrative tasks that you can perform as you operate your system. To point out the difference between required, recommended, and optional tasks, the section groups the tasks into categories. This section also includes a few topics that do not require an entire section of information.

Section 2. Monitoring System Activity

This section presents various ways to monitor activities on the system.

Section 3. Identifying Security Considerations

This section discusses the basic and fundamental aspects and concepts of system security. It presents the questions you need to ask concerning your security needs.

Section 4. Allocating System Files

This section discusses the characteristics of many of the system files and the system efficiency that is gained when you spread the system files across several disk families.

Section 5. Understanding the Disk System

This section introduces basic information about using disks on an A Series system and identifies tasks that can be performed.

Section 6. Protecting Disk Files

This section describes the cataloging feature and the archiving subsystem for A Series systems.

Section 7. Understanding the Print System

This section introduces you to concepts and terms unique to the Print System and the Remote Print System.

Section 8. Establishing the Printing Environment

This section identifies the tasks that you must accomplish to define the scope of your printing environment.

Section 9. Managing Processes at the Operating System Level

This section introduces methods that you can use to control the workload of your system.

Section 10. Managing the System Workload

This section discusses the methods available on A Series systems for controlling Work Flow Language (WFL) jobs through the use of job queues and priority ratings.

Section 11. Logging System Events

This section introduces you to the system and job logs, and presents the methods of extracting information from these log files.

Section 12. Billing for Use of System Resources

This section discusses how to capture billing information in the system log and how to generate reports that utilize that information.

Section 13. Analyzing and Reporting System Problems

This section discusses a method for analyzing system problems. It also discusses how to report problems or make suggestions to Unisys.

Related Product Information

***A Series Communications Management System (COMS) Capabilities Manual* (form 8600 0627)**

This guide introduces the Communications Management System (COMS), discusses the flexibility and efficiency of the system, describes the COMS architecture, and discusses specific features available to the COMS user. This guide is written for upper management, the site manager, and the programming staff.

***A Series Disk Subsystem Administration and Operations Guide* (form 8600 0668)**

This guide describes the logical structure and the operation of the disk subsystem. It explains disk subsystem concepts and terminology; explains the benefits and requirements of Cataloging, Mirrored Disk, and Memory Disk; and describes system messages displayed during directory management procedures. This guide is written for operations center managers, senior operators, and systems programmers.

***A Series Documentation Library Overview* (form 8600 0361)**

This overview describes the library of A Series software documentation. It also provides an explanation of titling conventions, the procedure for ordering documentation, and an introduction to online documentation and its role in A Series product documentation. This overview is written for all users of A Series systems.

***A Series Menu-Assisted Resource Control (MARC) Operations Guide* (form 8600 0403)**

This guide provides an overview of MARC, a description of the menu structure, and information on how to use help text, commands, security features, and Communications Management System (COMS) windows from MARC. The guide also explains how to run programs from MARC, how to customize MARC to meet user needs, and how to use MARC in a multinational environment. This guide is written for a wide audience, ranging from experienced system administrators to end users with no previous knowledge of MARC or A Series systems.

***A Series Memory Subsystem Administration and Operations Guide* (form 1169836)**

This guide presents an overview of memory architecture and memory management concepts. It explains how the memory subsystem functions and how to use and configure the memory subsystem. This guide is written for system administrators and operators.

***A Series Operating System Installation Guide* (form 8600 1021)**

This guide describes how to use the UTILOADER and LOADER programs, which are used to install the operating system when a system is first initialized or when the operating system is not functioning. This guide is written for system administrators and operators who might be responsible for installing the operating system.

***A Series Print System (PrintS/ReprintS) Administration, Operations, and Programming Guide* (form 8600 1039)**

This guide describes the features of the Print System and provides a complete description of its command syntax. This guide is written for programmers, operators, system administrators, and other interactive users of Menu-Assisted Resource Control (MARC) and CANDE.

***A Series Security Administration Guide* (form 8600 0973)**

This guide describes systems-level security features and suggests how to use them. It provides administrators with the information necessary to set and implement effective security policy. This guide is written for system administrators, security administrators, and those responsible for establishing and implementing security policy.

***A Series Software Release Installation Guide* (form 8600 0981)**

This guide tells you how to use the Simple Installation (SI) program to install a new software release on an established A Series system. The guide also contains specific installation instructions for the current Mark release. This guide is written for system administrators, operators, and others responsible for the installation of a new software release.

***A Series System Commands Operations Reference Manual* (form 8600 0395)**

This manual gives a complete description of the system commands used to control system resources and work flow. This manual is written for systems operators and administrators.

***A Series System Operations Guide* (form 8600 0387)**

This guide describes the basic concepts and procedures required to operate Micro A through A 6 systems and, more generally, all A Series systems. This guide is written for A Series operators, especially those with little or no experience.

***A Series Systems Functional Overview* (form 8600 0353)**

This manual presents an overview of the A Series systems and serves as a central source of information for these systems. This overview is written for both new and experienced users of A Series systems, and for anyone wanting an introduction to these systems.

Contents

Section 6. Protecting Disk Files

Section 7. Understanding the Print System

Section 10. Managing the System Workload

Section 11. Logging System Events

Section 12. Billing for Use of System Resources

Section 13. Analyzing and Reporting System Problems

Tables

Section 1
Identifying Initial Administrative Strategies

This section introduces you to the tasks that you are most likely to encounter as an A Series system administrator.

To use the procedures and recommendations in this guide, your hardware and system software must already have been installed. Whether or not installation has taken place, you can use this guide in two ways. You can read it to gain a better understanding of your administrative duties and to gain an idea of the kinds of decisions that you might need to make. However, system installation must be completed before you can perform any of the tasks described in this guide.

During system installation the following tasks are accomplished:

- Loading the Unisys software onto your halt/load unit
- Determining the unit numbers of the devices on your system
- Locating a USERDATAFILE and optionally adding usercodes to it
- Setting some basic master control program (MCP) options
- Initializing data comm
- Setting an automatic power-on and power-off schedule (A 1 through A 6 systems only)
- Setting up a memory dump file on disk
- Loading controlware files to the disk controllers, if necessary

For installation procedures, refer to the installation guide for your type of system. The various installation guides are among the A Series publications listed in the Bibliography at the back of this guide. You might find it useful to consult the installation guide for your system at this time. If you do, return to this guide afterwards and continue reading from this point.

This section distinguishes between required, recommended, and optional administrative tasks. Tables 1–1 through 1–3 list these tasks and direct you to where you can find more information about each one. Perform the required tasks immediately after system installation and before attempting to run user applications on the system.

Ideally, you should also perform the recommended tasks at this time, together with any optional tasks that you choose. Overlooking these tasks can result in your system running at less than its peak efficiency. However, if there is an urgent need to have your applications running without delay, you can perform the recommended and optional tasks at a more convenient time.

Required Administrative Tasks

Table 1–1 presents the administrative tasks that must be accomplished to run an efficient computer site.

Table 1–1. Required Administrative Tasks

Task	Pointer to Information about Performing the Task
Configuring data comm	If you plan to use only the stations and protocols in the default DATACOMINFO file, make sure the addresses of the terminals match the addresses in the default DATACOMINFO file. Refer to the software installation guide for your system for the description of the DATACOMINFO file. If you wish to add more stations and protocols, refer to "Setting Up Data Comm on Your System" in this section.
Installing applications software	For instructions on installing Unisys or third-party applications software, refer to the vendor documentation. Refer to your own documentation for installation instructions concerning applications developed in-house.
Controlling access to your system	There are many levels of securing your system. Section 3 of this guide, "Identifying Security Considerations," covers the more basic security issues. For in-depth discussions of security issues, refer to the *A Series Security Administration Guide.* The simplest method of securing your system is to stop individuals from entering the main operations room or accessing the operator display terminal (ODT). The next level of security is to require anyone accessing the system from remote terminals to use a usercode. Refer to the system installation guide for your system to obtain a brief account of adding usercodes to the Usercode Description file (actual file name is SYSTEM/USERDATAFILE). For more information about Usercode Description file, refer to the *Security Administration Guide.*
Setting up backup procedures	To be able to recover from an unusual situation, you should decide which disk files must be backed up to tape and how often a backup should occur. Refer to "Establishing Backup Procedures" in this section.
Determining how long to keep a system log	Because your system log can take up a large amount of disk space, it is advisable to develop a procedure for removing or transferring system log files. Refer to Section 11, "Logging System Events," for more information.

Recommended Administrative Tasks

Table 1–2 lists administrative tasks that should be performed at most sites, but need not be performed to have an operative system.

Table 1–2. Recommended Administrative Tasks

Task	Pointer to Information about Performing the Task
Configuring the Print System	The Print System functions without any changes to the delivered system. More information is presented about the printing system, including concepts, efficient operation, and customization of the printing environment, in Sections 7 and 8 of this guide.
Naming your disk families	The disks on your system arrived from the factory with family names already assigned to them. You can leave the preassigned family names intact or you can rename them if the names duplicate any family names you are using already. You may also want to rename some families to create any multidisk families you require. Before you move on to Section 5, "Understanding the Disk System," refer to any third-party vendor documentation to give you an idea about the family names your third-party applications may require.
Keeping records	Every computer site should keep records so there is historical information to help solve problems. This section contains helpful information under "Deciding Which Records to Keep."
Managing disk space	Since disk space is a critical commodity on a system and the inefficient use of disk space affects system performance, you should develop a plan to analyze the disk usage and institute a procedure to execute the actual disk cleanup. Refer to Section 5, "Understanding the Disk System," for information about managing disk space.

continued

Table 1–2. Recommended Administrative Tasks (cont.)

Task	Pointer to Information about Performing the Task
Handling the system work flow	The execution of processes (jobs and tasks) is controlled in two ways: by the operating system (also known as the Master Control Program or MCP), and by you. The MCP controls processes by using the factors of priority, scheduling, and suspension. Refer to Section 9, "Managing Processes at the Operating System Level, " for more details about these factors. In addition, most sites have programs that must be run daily, weekly, monthly, and/or yearly. Planning when these programs are to be initiated can be handled by creating a schedule that is followed by operators or by writing and executing sophisticated Work Flow Language (WFL) jobs in conjunction with the use of job queues. Refer to Section 10, "Managing the System Workload," for information about job queues. When executed, a WFL job initiates other jobs when a condition is met, a time period is arrived at, or other jobs have completed. Developing a WFL job is very helpful when you have a series of routine jobs to execute. For information about developing WFL jobs, refer to the *A Series Work Flow Language (WFL) Programming Reference Manual.*
Maintaining a Unisys documentation library	Your site should designate an area to house your basic Unisys documentation library. Such a library should contain all the manuals that your site needs. If well monitored, this library can serve a number of people without requiring that your site keep multiple copies of manuals. For descriptions of all the manuals in the A Series library, refer to the *A Series Documentation Library Overview.* You can also access this information online through the TEACH selection on the MARC main menu.

continued

Table 1–2. Recommended Administrative Tasks (cont.)

Task	Pointer to Information about Performing the Task
Contacting Unisys about problems	It is advisable to have one person responsible for contacting Unisys about problems with the system or the software. The individual responsible for reporting problems or making suggestions about new features and improvements should have a supply of User Communication Forms (UCFs). For more information, refer to Section 13, "Analyzing and Reporting System Problems."
Installing a new version of Unisys software or third-party applications	Installing new software usually does not require many people to be involved. If you have a large system, you might want one or more people to be responsible for preparing for and executing the installation of the new software. Installing new software should be done so that regular work is interrupted minimally or not at all. Refer to the *A Series Software Release Installation Guide* for information about using the SIMPLEINSTALL program to install new versions of Unisys software. For installation of new versions of third party applications, refer to the vendor documentation.

Optional Administrative Tasks

Table 1–3 lists administrative tasks that might be performed at your site if there is a need to increase the capability or performance of your system.

Table 1–3. Optional Administrative Tasks

Task	Pointer to Information about Performing the Task
Determining how users submit jobs	You should decide whether all batch jobs are to be initiated by an operator or whether programmers and other users should be allowed to initiate batch jobs on their own. For a discussion of the information that should be provided when a batch job is submitted to the operations staff, refer to "Submitting Jobs" in this section.
Allocating system software to disks	You can increase the efficiency of your system by spreading the system software across disks. To do so, you must first determine whether you have enough disk space. Section 4, "Allocating System Files," provides some general guidelines concerning allocation of files on disk families.
Enabling additional MCP Options	During the installation of your system you are asked to set some basic options that allow operations to run smoothly. Discussion about other MCP options and their usefulness is found in this section under "Choosing Additional MCP Options."
Generating system log reports	You might want to decide which of the information gathered in the system log should be seen in reports. Section 11, "Logging System Events," contains details about gathering information from your system log.
Limiting user disk space	The disk resource control (DRC) system allows a site to restrict disk space usage by usercode. When this mechanism is enabled you can do the following: • Control the maximum amount of space a user can have in use for permanent files on a family at any particular time. • Control the amount of space a user can use on an ongoing basis for permanent files on a family. • Control the total amount of temporary space a user task can have in use on the system at any particular time. For information about enabling the disk resource control (DRC) system, refer to the *A Series Disk Subsystem Administration and Operations Guide.*

continued

Table 1–3. Optional Administrative Tasks (cont.)

Task	Pointer to Information about Performing the Task
Suppressing messages on ODT and terminal	Your A Series system has a MSC (Message Control) MARC command that enables a user with SYSTEMUSER status to determine what messages appear on the ODT or remote terminal. You can use the MSC command to either suppress or show messages at user terminals based on a number of criteria such as usercode, chargecode, task name, and message type. For more information about using the MSC command, see Section 2, "Monitoring System Activity."
Billing customers for computer time	Your A Series system has a default system log that captures information about the tasks and conditions that your system has experienced. If you need to charge costs to outside customers or internal departments, you can indicate the information that needs to be captured in the log. Refer to Section 12, "Billing for Use of System Resources," for information on setting up such a charging system.
Using a supervisor program	A supervisor program (or startup program) is a specially designated code file that is run by the MCP immediately after a halt/load and before any other jobs are started. The purpose of the program is to configure your system to a specific state. You can either write your own supervisor program, or purchase the Unisys System Assistant program. For an explanation of how to develop a supervisor program, see "Developing a Supervisor Program" later in this section.
Using cataloging	Cataloging enables you to keep track of the backup copies of different versions of a file. Refer to Section 6, "Protecting Disk Files," for a discussion of cataloging.
Changing system memory factors	Sometimes a performance problem can be solved by adjusting the system's memory factors. For more information, refer to "Changing System Memory Factors" in this section.
Controlling in-house applications	To maintain production control, one person or a group of people should have the responsibility for installing in-house applications for production. Records should be kept to track who installed the program, who developed the program, when the installation was done, and which installation parameters were used, including the disk family where the program is located.

continued

Table 1–3. Optional Administrative Tasks (cont.)

Task	Pointer to Information about Performing the Task
Determining documentation needs for in-house applications	In order for your operators to be able to execute the programs at your site, a documentation standard should be developed to explain how and when a program should be run. This documentation should inform the operator about the data to be used and the output to be expected. If you are using third-party programs, their documentation might be all you need. If you are developing in-house applications, you should develop documentation forms to aid the operator when executing a given application. You should consider two types of documentation, one for batch jobs and one for online applications. If you have chosen to develop sophisticated WFL jobs, some of your requirements for documentation may not be necessary.

Setting Up Data Comm on Your System

If your system has more terminals than the default DATACOMINFO file supports or will have terminals that require protocols other than those provided in the default DATACOMINFO file, you must modify the DATACOMINFO file. Refer to the *A Series Data Communications Protocols Installation and Implementation Guide* and the *A Series Interactive Datacomm Configurator (IDC) Operations Guide* for information about modifying the DATACOMINFO file. The *System Operations Guide* gives procedures for installing the modified DATACOMINFO file.

Establishing Backup Procedures

As the system administrator, you must develop a policy to handle the backup and recovery of system and user files. If you do not do this, you run the risk of losing the only copy of these files in the event of a failure of the media on which they are stored.

Deciding Which Files to Back Up

You should first decide which files on your system are important enough to back up. Unisys suggests you make tape backups of the following files:

- SYSTEM/USERDATAFILE, if usercodes have been added
- SYSTEM/DATACOMINFO, if the default file is modified

The following are some questions you should ask as you decide what other files to back up and how often you should do so:

- Is the information in the file critical to daily operations?
- Does the information in the file change often?
- Could you reconstruct the information easily?

The FILECOPY utility greatly simplifies file backup. Some of the ways that FILECOPY can be used are as follows:

- To back up all files updated since a certain date and time
- To back up all files accessed since a certain date and time
- To back up all files created before a certain time
- To back up all files since the last run of FILECOPY
- To exclude or include certain file names

Refer to the *A Series System Software Utilities Operations Reference Manual* for information about using FILECOPY. Several third-party software houses have archiving programs that also create printouts of tape labels and keep a log of the backup tapes that have been created.

For more information about the Unisys archiving subsystem, see the discussion of using the archiving subsystem to back up and restore disk files in Section 6 of this guide.

Deciding the Extent and Storage of Backup Files

First you should decide how many generations of information to keep and where to store those generations. Many sites keep three generations of information available at all times. This method allows the site to fall back to another generation if the youngest generation of information cannot be used. You should analyze the needs of your site to determine how many generations of information should be kept.

Next you should determine where to keep those generations of information. Many sites have offsite storage to ensure that a fire or other disaster at the main site will not destroy the only copies of information. Some sites have locked areas for security purposes. Again, consider the needs of your site when making these decisions.

Deciding Which Records to Keep

Most sites find it appropriate to keep a log at the operator display terminal (ODT) of the system. Such a log is used to record problems that anyone has had when using the system and the peripherals. A log can be as simple as a blank sheet of paper or as complex as a company form. In any event, you should make sure that everyone on your staff knows what information should go in the log.

Reports about the names and characteristics of the files on your disks can be very beneficial. The FILEDATA utility creates such reports. Table 1–4 lists some of the reports, their characteristics, and their uses.

Table 1–4. FILEDATA File Reports

Report	Characteristics	Uses
ATTRIBUTE	Lists the files hierarchically with their file attributes.	Good way to document your critical files.
CODEFILEINFO	Lists the code files hierarchically and indicates which version of the compiler was used and any peculiar characteristics of the file.	Good way to keep track of current software.
FILENAMES	Lists the files hierarchically with certain file characteristics.	A way to see creation date, when a file was accessed last, the security class, and size of the files.
COPYDECK	Lists the files in a specific directory.	A primitive directory scanner.

You should also keep records of the names and versions of the software that is currently executing on your system. Knowing the current versions of the applications developed in-house is as important as knowing the current version of the Unisys operating system.

Keeping records about the versions of application software that run with corresponding versions of the operating system is also important. This information helps you if you must reload your software owing to an unusual situation.

You also might want to institute a procedure to keep I/O traffic statistics on each device, online response-time statistics, and transaction-count statistics. Determining the statistics that should be gathered depends on the activities of your site.

Choosing Additional MCP Options

The master control program (MCP) options enable you to customize the operating condition of your system. There are some options that every site finds useful, while other options have a more limited application.

Some options are set automatically when you start the system. To find out which options are already set on your system, enter *OP* in the Action field of any MARC screen, or enter *?OP+* at a Command and Edit (CANDE) control station.

To set or reset an MCP option, use the MARC *SOP* screen, accessed from the *SYS* menu, or enter *?OP+ <option name>* or *?OP- <option name>* respectively at a CANDE control station.

Table 1–5 lists the default options and the additional options that are suggested in the installation guides.

Table 1–5. Installation Options

Option	Meaning When Set
Option 2/TERMINATE	Abnormal terminations are processed normally instead of initiating a full memory dump.
Option 4/LPBDONLY	Printer output files are assigned to a printer backup disk by default.
Option 5/AUTORM	The MCP automatically removes the old disk file when a duplicate-file condition occurs.
Option 8/AUTORECOVERY	A halt/load is attempted after all system-fatal memory dumps, except when the system has a hung processor. Following the halt/load, the system reinitializes all network support processors (NSPs) running before the halt/load and restores the print server count and mix limit number to the values they had before the halt/load.
Option 12/AUTODC	Data comm is automatically initiated if it is not running when a message control system (MCS) performs a DCWRITE operation.
Option 14/CPBDONLY	All card punch files are assigned to a punch backup disk.
Option 16/CRUNCH	Unused space in the last area of a code file or printer backup file is automatically returned to the system when the file is closed.
Option 27/SERIALNUMBER	Inhibits the assignment of scratch tapes unless the serial numbers match. If the SERIALNUMBER system option has been set and a serial number has not been specified, a "<file name> REQUIRES" message is displayed, and the operator can respond with a DS (Discontinue), OU (Output Unit), or SN (Serial Number) system command. If the SERIALNUMBER system option is not set, the system uses any available scratch tape. For further information about using the OP (Options), DS, OU, and SN system commands, refer to the *System Commands Reference Manual*.

The MCP options listed in Table 1–5 permit you to use your system without further MCP option modification unless your system is to use any of the following capabilities: BNA network architecture, cataloging, or mirrored disks. Table 1–6 lists the BNA options, Table 1–7 lists the cataloging options, and Table 1–8 lists the mirrored disk option. The remaining MCP options, listed in Table 1–9, either give you additional opportunities to customize your system or help with debugging problems on your system.

Once you have decided which additional options to enable, remember to store this information either in a supervisor program or in a written log so that the same conditions can be reconstructed after a halt/load or a cold-start.

For more details about supervisor programs, refer to "Developing a Supervisor Program" later in this section.

For a complete listing of all the MCP options, refer to the *System Commands Reference Manual.*

Table 1–6 lists the MCP options that can be used if BNA is installed on your system.

Table 1–6. BNA Options

Option	When to Use
Option 40/NETRECOVERY	Enable this option if you are using BNA. When this option is enabled, the state of BNA is checked after the system is halt/loaded. If BNA was not running before the halt/load, BNA is not initiated. If BNA was running before the halt/load, all the BNA libraries and stacks are brought up.
Option 44/PORTDEBUG	Enable this option if you are using a diagnostic MCP and want port debug trace information to be written to the system log in the BNA debug log records. The captured information can be examined by using LOGANALYZER. However, note that use of this option may degrade system performance and use a significant amount of disk space. Normally the option should be disabled.

Table 1–7 lists the MCP options available if you are going to use the cataloging function of the system. For more information about cataloging, refer to Section 6, "Protecting Disk Files."

Table 1–7. Cataloging Options

Option	When to Use
Option 23/CATALOGING	Enable this option if you want to use cataloging. If you have enabled the OKTIMEANDDATE option, you can enable or disable the CATALOGING option before verifying the time and date. This setting becomes effective immediately. If you need to enable or disable this option after the initialization phase of the MCP, you must halt/load the system to have the setting take effect.
Option 28/ARCHIVING	Enable this option if you use the archiving function of the catalog subsystem. If this option is set and the catalog level of the system is greater than 0 (zero), an archive log is created. The name of the archive log is ARCHIVELOG/<date>/<time>, where <date> and <time> are the creation date and creation time of the file, respectively. If the archive log cannot be set up, the system automatically resets the option.
Option 45/USECATDEFAULT	Enable this option if you are using the cataloging function of the system and you want the USECATALOG file attribute to be set to TRUE by default.

Table 1–8 lists the MCP option that can be used if you are going to use the Mirrored Disk feature of the system. For more information about the Mirrored Disk feature, refer to "Mirrored Disk" in Section 5, "Understanding the Disk System."

Table 1–8. Mirrored Disk Option

Option	When to Use
Option 34/MIRRORING	Enable this option if you want to begin using disk mirroring after the next halt/load of your system. Enabling this option creates a Mirror Information Table (MIT) and other structures needed for disk mirroring. You cannot disable this option if any mirrored sets are currently present on the system. To disable this option, either close the mirrored sets or reduce the mirrored sets to single disks with the MIRROR RELEASE form of the MIRROR (Mirror Disk) system command.

Table 1–9 lists the MCP options that enable general-use functions and help in debugging your system.

Table 1–9. General Options

Option	When to Use
Option 1/OPEN	Enable this option if you want the system to generate a message each time a task opens a file.
Option 3/NOCHECK	Enable this option if the MCP forced a memory dump when it identified a FORGETCHECK condition and you do not want a memory dump every time this condition is experienced. Use this option when you cannot replace the current version of the MCP with another version.
Option 6/DIAGNOSTICS	Enable this option if you want the system to generate a message when a hardware unit is operating in a degraded mode.
Option 11/TRANSWARNINGS	Enable this option if you want the system to generate a message any time a file is opened and there is a risk of invalid data access and/or data corruption because of data transaction. The program code file is also marked to indicate that the opened file was at risk of invalid data access and/or data corruption. Refer to the *A Series I/O Subsystem Programming Guide* for information about hardware and software translation.
Option 13/NODUMP	Enable this option if you do not want memory dumps to be taken, but do want potential nonfatal dumps (those that do not end in a halt/load) to be noted at the ODT and in the system log. Conditions that cause a memory dump to end in a halt/load (a fatal dump) cause a halt/load even if this option is enabled.
Option 17/BACKUPBYJOBNR	Enable this option if you want print jobs to print in order according to job number rather than according to the number of lines in the print job. This option overrides the method of print request selection specified in a PS SELECTION command. For more information, refer to "Specifying How Print Requests Are Selected" in Section 8.
Option 19/NOFETCH	Enable this option if your Work Flow Language (WFL) jobs use the FETCH statement and you want to eliminate the need for the operator to enter an OK (Reactivate) system command to reactivate such jobs.
Option 20/RESOURCECHECK	Enable this option if you want a job to adhere to the limits on the type of tape and the number of tape units you set with the MQ (Make or Modify Queue) system command.

continued

Table 1–9. General Options (cont.)

Option	When to Use
Option 24/OKTIMEANDDATE	Enable this option if you want the operator to verify the time and date of the system after each halt/load. Do not enable this option if you are using the scheduled power up and down feature available on selected systems. Use the TR (Time Reset) system command to make time and date corrections before entering the TIMEOK response.
Option 26/LOGPOSITIONING	Enable this option if your system is experiencing tape reading errors such as lost blocks on tape. When this option is enabled, the MCP writes the positioning actions of tape parity retries, as well as the actual retries, into the system log. This information can be used to understand tape problems.
Option 35/DIAGNOSTICDUMP	Enable this option if you need additional memory dumps to aid in diagnosing a program problem.
Option 37/FILESATURATION	Enable this option if you want the system to generate a message that the last row of an application or compiler code disk file is being allocated. The operator is not notified of last row allocation for files created by the MCP such as the system log file or library maintenance files. Remember that this notification does not necessarily mean that the file is almost full. Some rows might have been skipped if the file was written randomly. If a file has the FLEXIBLE file attribute enabled, the message is issued only when row 1000 of the file is allocated. Since most files use the default value of TRUE and few files contain 1000 rows, this warning message seldom appears.
Option 38/EOTSTATISTICS	Enable this option if you want the system to display end-of-task (EOT) statistics on the ODT as each task ends. The message displays the elapsed time, the processor time, and the I/O time in hours, minutes, and seconds.
Option 39/PATHBALANCING	Enable this option if you have a multiple-path A 1 through A 9 system and want to have dynamic path balancing. Dynamic path balancing distributes disk I/O operations more evenly across busy multiple-path disk systems. Static path allocation is always used for tapes, since the order in which the data is transferred is crucial.

Using Structure Caches

The operating system by default maintains caches for a number of data structures. These caches can produce significant performance improvements, especially for task initialization and termination, and when disk files are opened. There is a cost for the use of this new feature, in that the caches use additional system memory. You can use the STRUCTURECACHE (Cache Maintenance) system command to disable this feature at your site and prevent allocation of system memory for these caches.

For more information on the STRUCTURECACHE system command, see the *System Commands Reference Manual.*

Submitting Jobs

If you decide that all batch jobs are to be initiated by the system operator, then you should determine which forms should be made available for users so they can submit requests for jobs. You should also determine who will accept those job requests. Such forms usually ask for the following information:

- The name of the program to be run
- The options to be set, if any
- The data to be used
- The storage location of the data
- The destination of the output
- The name of the person to call for information
- The date and time the output is needed

Developing a Supervisor Program

A supervisor program is a program that the MCP runs immediately after a halt/load and before it starts any other jobs. The use of a supervisor program is optional, and its purpose is to configure your system to a specific state. Unisys offers a supervisor program called System/Assistant, available separately.

Use a supervisor program if you want to perform certain tasks every time the system is halt/loaded. Alternatively, you can use such a program if you want your operator to have a simple way to return to normal operating conditions after an unusual condition forces the operator to change the MCP options.

A supervisor program is also helpful when you are trying to solve a throughput problem on your system. System options are often changed during the course of the day and might not get returned to their normal settings. Running a supervisor program allows you to determine if the throughput problem was caused by inappropriately set system options.

Some tasks a supervisor program can do are the following:

- Ensure that MCP options are set correctly.
- Set up job queues.
- Start site-specific programs.

One way to develop a supervisor program is to use the DCALGOL compiler and the DCKEYIN statement. This statement enables you to send system commands to the operating system and receive corresponding responses back from the operating system. For the program to function as a supervisor program, it must be run under a privileged usercode or be designated as a privileged program; in addition, it must be located on the halt/load unit and be designated as a supervisor program by the CS (Change Supervisor) system command.

The DCKEYIN statement limits the programmer to an array of up to 256 words. That limitation affects large and very large sites that often have a response that exceeds 256 words. At these sites, calls to the SETSTATUS and GETSTATUS functions of DCALGOL are the only way to receive the longer responses.

When you develop your supervisor program, it is advisable to give it a simple name that relates to its function. STARTUP or SUPERVISOR are good names. If STARTUP is used, the operator would enter ??RUN STARTUP anytime the program needed to be executed other than at a halt/load.

Changing System Memory Factors

There are two system commands that enable you to adjust memory factors. The ASD (Actual Segment Descriptor) system command adjusts the size of the ASD table; a system running programs that use relatively few large arrays needs a smaller ASD table, and a system with programs that use many small arrays needs a larger ASD table.

The SF (Set Factors) system command allows you to modify the memory overlay rate, specify the minimum amount of memory that must be available on the system, and control other algorithms that modify the way memory functions. The following factors can be set:

- OLAYGOAL sets the percentage of overlayable memory in the system that is to be overlaid every minute.
- AVAILMIN determines the minimum amount of total memory that is to be available at all times.
- FACTOR determines when enough memory is available for a task to be initiated.
- MEM PRIORITY FACTOR determines the amount of time that data belonging to a higher priority program remains in memory before being overlaid to make space for a lower priority program.
- BUFFERGOAL allows you to modify the default value of the BUFFERSIZE file attribute to something other than the value chosen by the system.
- OLAYSATURATION determines the level of processor usage which can be used for overlay activity.
- OLAYCHANGE is used to measure the overlay activity.

For a discussion of when you might want to use the SF system command, refer to the *A Series Memory Subsystem Administration and Operations Guide.*

Section 2
Monitoring System Activity

One of your main duties as a system administrator is to extract pertinent information from the system, analyze that information, and then make decisions based on that information. To carry out these duties, you need to determine how you can use your system, how it performs under various conditions, and where you can make changes, if necessary.

Information of this type is obtained by monitoring various events and processes. The knowledge gained by monitoring your system also allows you to increase or fine-tune performance for maximum efficiency and to detect any degradation in performance caused by procedural changes or hardware problems.

Many factors affect system performance on an A Series system. Monitoring all these factors manually would be impossible. Unisys provides the following software and tools for logging, retrieving, displaying, and printing information about pertinent events:

- U (Utilization) system command
- Activity Reporting Systems (BARS) utility
- System Management Facility II (SMFII)
- SYSTEMSTATUS, GETSTATUS, and SETSTATUS intrinsic routines
- LOGGER and LOGANALYZER utilities
- Communications Management System (COMS) Statistics facility
- DBANALYZER utility
- DMMONITOR utility
- Message Suppression

U (Utilization) System Command

The U (Utilization) system command provides current statistics about system usage, including central processor unit (CPU), I/O, MCP, and total system use. This command can be part of the automatic display mode (ADM) display for periodic checks by the operator. If you want a permanent record of these statistics for trend analysis, you can obtain such a record by using the System Management Facility II (SMFII).

The U command divides the information it displays into processor and I/O categories. Each component of the display is given as the percentage of elapsed time since the last time interval. The time interval, which can vary, is controlled by the SBP (System Balancing Parameters) system command.

Following is an example of the display produced by the U command:

```
------------------------- SYSTEM UTILIZATION -------------------------
------- CPM STATISTICS ------------  ---------- IO STATISTICS ----------
 USER      = 23% INITIAL PBIT =  2%  USER    =  1  IO/SEC  (  1 KB/SEC)
 IO FINISH =  1% OTHER PBIT   =  Ø%  MCP     =  7  IO/SEC  (  7 KB/SEC)
 MCP       =  4% TRUE IDLE    = 7Ø%  DATACOM =  2  IO/SEC  (  2 KB/SEC)
 SEARCH    =  Ø% FALSE IDLE   =  Ø%  TOTAL   = 11  IO/SEC  (  7 KB/SEC)
                                           11   IO - INTERRUPTS/SEC
```

The first part of the U command display, labeled CPU or CPM, gives information about processing utilization, which lists the percentages of processor time spent on each of eight different task categories. These categories are discussed in the following table.

Table 2–1. U Command Display—Processor Data

Category	Meaning
USER	Indicates the processor time spent in user stacks. This category includes user programs, job stacks, and library maintenance. Independent runners (MCP procedures initiated as independent processes) and time charged to accounts such as IOFINISH and INITIALPBIT are not included.
I/O FINISH	Indicates the processor time spent handling I/O completion.
MCP	Indicates the processor time used by the MCP in managing the system.
SEARCH	Indicates the processor time spent in connection with searching stacks.
INITIAL PBIT	Indicates the processor time spent reading arrays and code into memory for the first time.
OTHER PBIT	Indicates the processor time spent reestablishing arrays and code after they have been overlaid, and time spent overlaying existing data to make room for new data. Time spent searching stacks during the overlay process is not included.
TRUE IDLE	Indicates processor time spent in an idle state that is not false idle.
FALSE IDLE	Indicates processor time spent idle while overlaid data is being transferred by the I/O subsystem. The false idle percentage indicates that overlay is causing some tasks to experience increased turnaround time. It is not necessarily a measure of the amount of existing work that would be performed if overlay were not a factor; use of working set algorithms can force overlay and lead to false idle time.

The second part of the display, labeled IO, gives information about input/output utilization. Table 2–2 lists the contents of this part of the display.

Table 2–2. U Command Display – Input/Output Data

Category	Meaning
USER	Indicates the average number of user I/O operations per second and the number of kilobytes of data transferred per second by user I/O operations.
MCP	Indicates the average number of MCP I/O operations per second and the number of kilobytes of data transferred per second by MCP I/O operations, which include such functions as overlay and logging.
DATACOM	Indicates the average number of data comm I/O operations per second and the number of kilobytes of data transferred per second by the data comm I/O operations, which transfer messages between the NSP and the host.
TOTAL	Indicates the total system I/O activity: the total number of user and MCP I/O operations (plus or minus 1).
I/O INTERRUPTS	Indicates the average number of I/O interrupts per second.

Activity Reporting Systems (BARS) Utility

The BARS utility does real-time monitoring of system performance and displays performance statistics as numeric values and bar graphs. The items displayed are those concerning current CPU, memory, I/O, and disk use. The display is updated periodically and automatically contains only those items that apply to the system on which BARS is running.

You can either select from a comprehensive list the items for the BARS utility to monitor or use the default list supplied with the program. Newly created lists can be saved for later reuse.

BARS provides an alternative to SMFII in monitoring the current activity on the system. You can use BARS interactively or in batch mode. In batch mode, the utility writes the performance data to the file specified when the utility is started. You then provide an application program to view the file contents. BARS can be accessed through Menu-Assisted Resource Control (MARC) menus or through the Command and Edit (CANDE) system.

Additional information on BARS can be found in the *A Series System Software Support Reference Manual.*

System Management Facility II (SMFII)

The System Management Facility II (SMFII) monitors and provides information on four areas of system performance:

- Hardware performance
- Software performance
- Workload characterization
- System utilization

SMFII creates and maintains a database containing the statistics that have been gathered on these four performance areas. The collected data is accessed, and reports generated and formatted, with the QUERY program of SMFII.

SMFII consists of three major components:

- The Site Management System is used for monitoring the operating condition of the hardware and software components of the system. The Site Management System contains two modules:
  - The hardware module contains information about the occurrence and nature of faults in mainframe and peripheral devices detected by the system during normal operation, such as I/O errors.
  - The availability module contains information about the occurrence, nature, extent, and resolution of events that interfere with total system use, such as equipment malfunctions and operations problems.
- The System Resource System is used for measuring the system workload and the utilization of system resources. This component also contains two modules:
  - The workload module contains data about the programs run on the system and about user resource demands, such as program file access.
  - The utilization module contains information about program use of hardware and software resources, such as memory management and data communication statistics.
- SMFII/QUERY is used for the analysis of the collected data and the generation of reports. QUERY accepts report specifications, analyzes data, and produces graphic and statistical reports in a variety of formats. QUERY can be used interactively or in batch mode.

The Site Management System and SMFII/QUERY are released to all customer sites as a part of the standard system software; the System Resource System is a separately priced item.

For detailed information about SMFII and its component programs, refer to the *A Series System Management Facility II (SMFII) Resource Management Operations Reference Manual* and the *A Series System Management Facility II (SMFII) Query Operations Guide.*

SYSTEMSTATUS, GETSTATUS, and SETSTATUS Intrinsic Routines

SYSTEMSTATUS is an intrinsic MCP routine (a system function within the MCP) that gathers many different types of information concerning the activity and environment of the system. SYSTEMSTATUS returns groups of related information. You use SYSTEMSTATUS when you want comprehensive information concerning a particular area of the system, such as job queues or hardware configuration. The *A Series SYSTEMSTATUS Programming Reference Manual* describes this intrinsic routine and explains how to use it.

GETSTATUS and SETSTATUS are also intrinsic routines within the MCP. The GETSTATUS routine retrieves information about the job or task mix, peripheral and disk unit status, MCP and configuration status, and the files in the disk directories. SETSTATUS provides the system interface for an object program that controls the mix, unit, and operational functions of the MCP.

Both GETSTATUS and SETSTATUS routines can be called only by a DCALGOL message control system (MCS) or a DCALGOL user program executing under a privileged usercode or with privileged program status. An exception to this restriction is the GETSTATUS directory query function. It can be used by someone with a nonprivileged usercode to examine data about files accessible to that usercode.

GETSTATUS and SETSTATUS are used by performance-monitoring utility programs such as BARS and are available for use by sites whose users want to write their own utility programs. The *A Series GETSTATUS/SETSTATUS Programming Reference Manual* contains information on these two intrinsics.

LOGGER and LOGANALYZER Utilities

LOGGER and LOGANALYZER are utility programs for examining system log files. SYSTEM/LOGANALYZER is a program that analyzes operations data and provides a history of beginning and ending of task times and file opening and closing times.

LOGANALYZER also produces a report consisting of all SYSTEM/SUMLOG entries that correspond to parameters that you set. LOGANALYZER extracts the specified types of entries from the log and formats them for display on a screen, for printing, or for writing to a file. The LOGANALYZER utility examines any log file that you specify. If no log file is specified, LOGANALYZER analyzes the current SUMLOG file. You can also limit the LOGANALYZER search to entries in the log file that were made during a particular period.

The LOGGER utility generates reports that aid in the analysis of system performance and utilization. It extracts user-selected data from the SUMLOG files and can summarize, total, average, and sort the data to produce a report. LOGGER is less powerful and flexible than SMFII, but is useful for producing charging and billing information. LOGGER is best suited for generating reports about system performance, whereas LOGANALYZER is more useful for providing information about individual job runs. Both LOGGER and LOGANALYZER are described in the *System Software Support Reference Manual.*

Communications Management System (COMS) Statistics Facility

You can use the COMS Statistics facility to help you fine-tune COMS and eliminate bottlenecks in processing transactions. The Statistics facility can evaluate measurements from the COMS transaction trail, which is where the performance data for direct windows is written.

The Statistics facility is run through the Statistics window provided by COMS. The Statistics window has two main functions:

- Enables you to view COMS performance dynamically, online
- Initiates the generation of COMS statistics reports, online or printed

COMS Statistics are available in three different forms:

- Online screens
- Printed reports, standard or user-written, by means of the Extended Retrieval with Graphic Output (ERGO) utility
- Library calls

For information on using library calls to access COMS Statistics, see the *A Series Communications Management System (COMS) Programming Guide*. The *A Series Communications Management System (COMS) Configuration Guide* provides a more detailed description of COMS Statistics.

DBANALYZER Utility

The DBANALYZER utility is useful if you are running either a Semantic Information Manager (SIM) or Data Management System II (DMSII) database on your system. The utility provides a convenient method by which the database administrator can analyze the current implementation and status of an existing SIM or DMSII database.

For information about the DBANALYZER utility, refer to the *A Series DMSII Utilities Operations Guide.*

DMMONITOR Utility

The DMMONITOR utility enables you to interactively monitor the status and statistics of an active DMSII database. In addition, this utility permits you to make dynamic modifications of selected database options and parameters, and allows you to sample database information and store that information in a disk file for later offline analysis.

Designed for ease of use, DMMONITOR is compiled for each database using the DMALGOL compile-time facility and, by default, is given the name MONITOR/<database name>.

DMMONITOR uses the MultiLingual System (MLS) to obtain text for its messages. The MLS translates output messages into other languages. The language available in the released product is English.

For a more detailed description of the DMMONITOR utility, refer to the *A Series DMSII Utilities Operations Guide*.

Message Suppression

Message suppression is available through the MSC (Message Control) command of MARC and enables any user with SYSTEMUSER status to control what messages appear on the ODT and remote terminals.

The MSC command can be used to either suppress or show messages that are sent to the ODT and to remote terminals. This feature is especially useful for eliminating the copy messages from library maintenance jobs at the ODT, as well as messages from specific programs that the operator need not monitor.

Messages that are suppressed are not lost, but continue to appear in the printed job summaries. They can also be displayed on the ODT by using the *MSG ALL* system command.

Your site can dynamically suppress or display messages

- By message type
- By a unique characteristic, such as charge code, task name, or usercode

In addition you can control if messages are suppressed or displayed at the ODT, at a remote terminal, or at both locations.

You can use the MSC command to suppress or show messages in any of the following ways:

- By entering MSC commands on the Action line of any MARC screen
- By selecting options on MARC screens
- By loading one or more data files of MSC commands that you have created

  You can arrange to have data files of MSC commands loaded automatically whenever a halt/load occurs, or you can load them manually as needed.

  You can have more than one file of MSC message suppression commands to suit different situations. You might want to suppress certain messages during daily operations or at system initiation, and suppress a different set of messages during nightly or weekend processing. All MSC commands not specifically saved in separate files are erased when a halt/load occurs.

Note: The MSC command enables you to suppress messages that warn of potentially disastrous situations such as errors in copying that occur during a backup, or disk errors during a file copy operation. For that reason, take care when you and your operators determine what messages are to be suppressed and where the suppression should occur.

For basic information about using the message suppression feature, refer to the *System Operations Guide*. For information about creating a file of MSC commands and creating a default file of MSC commands for use after a halt/load, refer to the *MARC Operations Guide*.

Section 3
Identifying Security Considerations

The plan you devise for the security of your system and its data resources is dependent upon the needs and uses at your particular site. This section is designed as an introduction to security issues before you move on to the *Security Administration Guide*. This section presents you with some key questions to answer concerning your security needs and discusses some of the methods of implementing security on your system.

One method for making a computer system secure is by controlling user access to the system and its components. That is, you allow only certain people access to the computer. Consequently, all jobs must be channeled through one of these people. To apply this method of security, you must make provisions for controlling physical access to the computer room, tape libraries, and system terminals. If you plan to secure the system in this way, consider your requirements as early in the planning stage as possible.

You also have options for controlling security through various software mechanisms. Unisys supplies the means for security control through software; for example, there are network security features, compiler security features, and an auditing feature.

You can also use the Communications Management System (COMS) to define security measures for direct windows through the COMS configuration file (CFILE), through special processing items, and through direct-window programs. COMS directly controls access to various parts of the network through the use of authorized usercodes and authorized stations.

Each of these options is discussed briefly later in this section. Whether you implement any of these features depends on the security requirements for your site.

Access Control

Controlling access to the resources of the computer system is the fundamental tool available to the security administrator. A Series systems have two forms of access control: physical system access control and system resource access control.

Controlling Physical System Access

Physical access control involves controlling access to the physical components of the system, such as the processing unit, the tape drives, and the operator display terminals (ODTs).

Controlling physical access involves the following:

- Protecting the system and its component parts against deliberate damage
- Controlling access to devices that can be used to communicate with the system, such as remote terminals and ODTs
- Protecting storage media, such as tape reels and disks

Controlling System Resource Access

System resource access is access to files and to the computing power of the system. An individual has gained access to the system resources when he or she is able to carry on a dialog with the system beyond the log-on process.

Table 3–1 lists the three levels of resource access control.

Table 3–1. Resource Access Security Levels

Control Level	How It Works
Minimal security	Anyone logged on to the system has access to all files and resources. No security-related auditing is done. There is no formal security policy for users of the system.
Moderate security	Users are granted varying rights, depending on their needs. Unlimited access is reserved to those granted privileged status. Occasional auditing of security-related user activities is done. There is no formal security policy for users of the system.
High security	A security administrator is placed in charge of establishing and maintaining system security. This person is the only one allowed by the system software to define users on the system, to assign access privileges to users, and to use certain system commands that configure or maintain software security enforcement.

Controlling User Access

The first control of access to system resources is user identification based on a usercode. The system responds only to those authorized to use the system. The user also might have to enter a password that authorizes use of the usercode. If the usercode (and, if required, the password) is defined in the USERDATAFILE of the system, then the user is granted access to the system.

Note: *An important exception to the log-on procedure involves ODTs. These terminals accept input without requiring a user to log on to the system. For this reason, physical access to an ODT might need to be restricted.*

The session for the user runs under that usercode until the user logs off. The usercode is the basis for file ownership and can have other access rights associated with it, such as privileged user status.

It is possible for an individual to run a session that does not have a usercode associated with it. Such a user is very limited in the ways he or she can interact with the system.

Accesscodes provide an additional level of user security. They are similar to usercodes, have one associated password, and serve as a second log-on procedure after the usercode/password log-on. Accesscodes are also used for additional file security.

Usercodes, passwords, and accesscodes are created through the MU (Make User) system command or through the MAKEUSER utility program. The MU system command is enabled by default; anyone with access to the ODT can define a user. However, the MU system command can be disabled through the MAKEUSER utility. To access the MAKEUSER utility a user must have privileged or security administrator status.

Information about the MAKEUSER utility can be found in the *Security Administration Guide* and information about the MU system command can be found in the *System Commands Reference Manual.*

Privileged User Status

To perform certain necessary system functions such as archiving files on a daily basis, you can designate a few selected users as being *privileged.*

Except for some restrictions, a privileged user can access any file and run any program on the system, regardless of the file security designation. Privileged users can also use operating system procedures such as GETSTATUS and SETSTATUS. For accountability purposes, Unisys strongly recommends that the smallest possible number of persons be given privileged status.

SYSTEMUSER Status

The Menu-Assisted Resource Control (MARC) interface adds one level to the normal privileged and nonprivileged status – that of SYSTEMUSER. A SYSTEMUSER has operator privileges and capabilities and has access to functions that affect system operation. Having SYSTEMUSER status does not make a usercode privileged, and vice versa, but a usercode can be designated as both privileged and SYSTEMUSER.

Privileged Program Status

Programs can be marked as privileged, meaning that the programs have the same status as a privileged user and can access all files and privileged system functions. The PU (Privileged User) option of the MP (Mark Program) system command is used to give a program privileged status. Nonprivileged users can run these programs if they are specifically given access to the file through file security attributes.

Tasking Program Status

Programs can be marked as having tasking status, meaning that the programs have a security status that allows the programs to perform some functions that otherwise can done only by an MCS program. The TASKING option of the MP system command is used to give a program tasking status.

If you assign the TASKING option to a program that has TADS capability, and the system is running with security class S1 or S2, a warning message is displayed.

Refer to the *Task Attribute Programming Guide* for more information.

Controlling File Access

File access is controlled by file attributes, through which the owner of the file defines the level of security each file is to have. The owner can specify which users can access the file and the type of access each user can have. The following file attributes provide various security features:

- SECURITYTYPE. This attribute specifies who can access the file, for example, privileged or nonprivileged users.
- SECURITYUSE. This attribute defines the way in which a file protected by security can be used, for example, read-only access or read/write access.
- SECURITYGUARD. This attribute names the guard file to be used if the value of the SECURITYTYPE attribute of the file is either GUARDED or CONTROLLED. You can create a guard file by using the GUARDFILE utility.

For a list of the values you can give to the SECURITYTYPE and SECURITYGUARD file attributes, refer to the *A Series File Attributes Programming Reference Manual.*

A guard file describes the access rights of various users and programs to a disk file, tape volume, or database. Users are identified by usercodes and, optionally, by accesscodes; programs are identified by file name and family name. Both users and programs can be further restricted to access only when running certain programs or when running under certain usercodes, respectively.

When access to a file is controlled by a guard file, every attempt to open that file causes the operating system to examine the access rules in the guard file before granting or denying access to that file. If the guard file cannot be found, the file is treated as a private file. Guard files are created by using the GUARDFILE utility, described in the *A Series Security Features Operations and Programming Guide.*

InfoGuard Security Enhancements

InfoGuard provides enhancements and additions to security for the A Series systems. The InfoGuard features increase the level of system security and simplify the task of configuring a secure system.

Some of the features that InfoGuard provides are described in the following text. The *Security Administration Guide* and the *Security Features Guide* provide more detailed information.

System-Enforced Security Administrator Status

Access to security-sensitive functions can be denied to anyone who is not specifically designated as a security administrator in the USERDATAFILE. A user who has security administrator status can perform such functions as interrogating or modifying the USERDATAFILE, and setting or altering the system logging options by means of the LOGGING (Logging Options) system command.

The session for the user runs under that usercode until the user logs off. The usercode is the basis for file ownership and can have other access rights associated with it, such as privileged user status.

It is possible for an individual to run a session that does not have a usercode associated with it. Such a user is very limited in the ways he or she can interact with the system.

Accesscodes provide an additional level of user security. They are similar to usercodes, have one associated password, and serve as a second log-on procedure after the usercode/password log-on. Accesscodes are also used for additional file security.

Usercodes, passwords, and accesscodes are created through the MU (Make User) system command or through the MAKEUSER utility program. The MU system command is enabled by default; anyone with access to the ODT can define a user. However, the MU system command can be disabled through the MAKEUSER utility. To access the MAKEUSER utility a user must have privileged or security administrator status.

Information about the MAKEUSER utility can be found in the *Security Administration Guide* and information about the MU system command can be found in the *System Commands Reference Manual.*

Privileged User Status

To perform certain necessary system functions such as archiving files on a daily basis, you can designate a few selected users as being *privileged.*

Except for some restrictions, a privileged user can access any file and run any program on the system, regardless of the file security designation. Privileged users can also use operating system procedures such as GETSTATUS and SETSTATUS. For accountability purposes, Unisys strongly recommends that the smallest possible number of persons be given privileged status.

SYSTEMUSER Status

The Menu-Assisted Resource Control (MARC) interface adds one level to the normal privileged and nonprivileged status – that of SYSTEMUSER. A SYSTEMUSER has operator privileges and capabilities and has access to functions that affect system operation. Having SYSTEMUSER status does not make a usercode privileged, and vice versa, but a usercode can be designated as both privileged and SYSTEMUSER.

Privileged Program Status

Programs can be marked as privileged, meaning that the programs have the same status as a privileged user and can access all files and privileged system functions. The PP (Privileged Program) system command is used to give a program privileged status. Nonprivileged users can run these programs if they are specifically given access to the file through file security attributes.

Controlling File Access

File access is controlled by file attributes, through which the owner of the file defines the level of security each file is to have. The owner can specify which users can access the file and the type of access each user can have. The following file attributes provide various security features:

- SECURITYTYPE. This attribute specifies who can access the file, for example, privileged or nonprivileged users.
- SECURITYUSE. This attribute defines the way in which a file protected by security can be used, for example, read-only access or read/write access.
- SECURITYGUARD. This attribute names the guard file to be used if the value of the SECURITYTYPE attribute of the file is either GUARDED or CONTROLLED. You can create a guard file by using the GUARDFILE utility.

For a list of the values you can give to the SECURITYTYPE and SECURITYGUARD file attributes, refer to the *A Series File Attributes Programming Reference Manual.*

A guard file describes the access rights of various users and programs to a disk file, tape volume, or database. Users are identified by usercodes and, optionally, by accesscodes; programs are identified by file name and family name. Both users and programs can be further restricted to access only when running certain programs or when running under certain usercodes, respectively.

When access to a file is controlled by a guard file, every attempt to open that file causes the operating system to examine the access rules in the guard file before granting or denying access to that file. If the guard file cannot be found, the file is treated as a private file. Guard files are created by using the GUARDFILE utility, described in the *A Series Security Features Operations and Programming Guide.*

InfoGuard Security Enhancements

InfoGuard provides enhancements and additions to security for the A Series systems. The InfoGuard features increase the level of system security and simplify the task of configuring a secure system.

Some of the features that InfoGuard provides are described in the following text. The *Security Administration Guide* and the *Security Features Guide* provide more detailed information.

System-Enforced Security Administrator Status

Access to security-sensitive functions can be denied to anyone who is not specifically designated as a security administrator in the USERDATAFILE. A user who has security administrator status can perform such functions as interrogating or modifying the USERDATAFILE, and setting or altering the system logging options by means of the LOGGING (Logging Options) system command.

Consult the *Security Administration Guide* for information about enabling security-administrator status, and for a complete list of functions that are restricted to the security administrator.

Password Aging

InfoGuard enables you to designate a "lifetime" and a warning period for passwords. When a password reaches a certain age, the user is warned that the password will expire in so many days. At the end of that time, the password can no longer be used to access the system.

Users whose passwords are about to expire can either generate a new password for themselves or apply to the security administrator for a new password.

Password Generation

You can require all users to have passwords that were generated by the computer. A MARC password screen provides an easy-to-use interface to the system, which generates a random, pronounceable password 7 to 11 characters long.

Simplified Security Administration

With the SECOPT (Security Options) system command, InfoGuard gives you a way to set a large range of system security options. If you like, you can assign appropriate values to all these options by simply designating a class of security for your system. Use of the SECOPT command is further simplified by the availability of the MARC *SECOPT* screen, which offers all features of the SECOPT command on an easy-to-use screen.

Tape Security

InfoGuard lets you implement tape security with tape volumes that have associated with them a usercode and certain attributes: SECURITYTYPE, SECURITYUSE, and SECURITYGUARD. Consequently, a tape volume can be

- Owned by a user
- Designated as a public or private volume
- Designated as having different kinds of access with the SECURITYUSE attribute
- Guarded with a guard file

Selective Logging

The LOGGING (Logging Options) system command provides a method of designating the activities to be recorded in the SYSTEM/SUMLOG and the job log. The availability of a series of MARC screens that offer all the features of the LOGGING command in an easy-to-use format further simplifies the use of the LOGGING command.

Security Workstation

With this feature you can designate a station to act as a security station. All system security messages are routed to the security workstation, allowing for easier monitoring of security-related activities.

C2 Security

Unisys A Series systems running standard software and employing InfoGuard have been certified at the C2 controlled-access protection level by the Department of Defense National Computer Security Center.

C2 certification for a system means that the system can be operated in a fashion that meets the National Computer Security Center requirements for class C2 computer security.

To meet the requirements for class C2 certification, as described in the Department of Defense Trusted Computer System Evaluation Criteria DoD 5200.28-STD, a system must have features such as the following:

- Identification of individual users through a log-on procedure
- Accountability of action at the level of individual user
- Secured reuse of system resources such that no user gains unauthorized access to data

If your system is to be run on an environment intended to meet the requirements for class C2 operation, consult the *Security Administration Guide* for further information.

Network Security

BNA network architecture has a distributed approach to security. In a BNA network, each host for an A Series and compatible computer system operates independently of the other hosts and controls its own resources. A host governs remote user access by maintaining a list of allowable remote usercode/hostname pairs in its USERDATAFILE. Each request for service from a remote user is checked against this list and subjected to the same degree of security checking as local requests.

In addition to controlling user, system, and file access, the host also controls access to system functions and to such resources as processor time, peripheral use, and file storage space. The host can expand or revoke access privileges at any time without notifying other hosts in the network.

Compiler Security

Three of the specialized languages offered by Unisys (DCALGOL, DMALGOL, and NEWP) contain language extensions that allow a program to access internal system functions and data files. You might want to limit access to the compilers for these languages.

A user does not need privileged status to use the compilers for these languages unless the compilers have been secured at that site. Compiler-generated programs that are not given privileged status will run as long as they do not try to access privileged system functions or files. However, if they are given privileged status, the programs can access some privileged system functions.

The NEWP compiler checks the potential effect on the system of program statements as it generates object code. The compiler specifically checks for *unsafe statements*. These are statements that, if used incorrectly, might cause a program to access memory areas outside the normal addressing environment of the program, or that would cause the system to halt. If the NEWP compiler finds unsafe statements, it compiles the program but marks it as unsafe.

The system does not execute a program that is marked unsafe unless the program has been installed by the system operator; for example, a new operating system, a system library, or a stand-alone program such as the LOADER would be marked unsafe, but executed by the system. The MCP and certain system libraries are the only routinely executed programs marked unsafe by NEWP.

To prevent any security problems from arising, you might want to limit access to the DMALGOL, DCALGOL, and NEWP compilers. You can do so by any of the following methods:

- Limit the number of privileged users on the system to a very small, controlled group.
- Physically remove the compiler from the system; return it only long enough to perform scheduled compilations, and then remove it again.
- Designate the compiler files as GUARDED or CONTROLLED and use a guard file to list the persons who can access them.

COMS Security Controls

The Communications Management System (COMS) enables you to limit the number of processes a COMS user can run simultaneously and the total number of processes that can be submitted from COMS sessions at a given time.

The COMS windowing capabilities enable a user at a single terminal to have many MARC and CANDE sessions in progress at the same time. Whenever a user opens a new CANDE window dialog, COMS creates a new CANDE session, which is accessed through that dialog. By limiting the number of window dialogs that are allowed to a user, you also indirectly limit the number of processes the user can run simultaneously.

You can use COMS to define security measures for the window dialogs allowed for each user by controlling access to

- Stations
- Windows
- Transaction codes
- Security messages

Station, Window, and Transaction Code Security

Station security involves assigning a valid-station list to a usercode to restrict the locations where the usercode can be used.

Window security involves assigning a valid-window list to a usercode to restrict the windows that can be opened with the usercode.

Transaction code (trancode) security involves restricting access to certain transaction codes. Because you use trancodes to perform some transactions within an application program, by restricting trancodes you can restrict the degree of transaction processing that the sender can perform. With COMS you can restrict access to trancodes by the following entities:

- Programs
- Stations
- Usercodes

You can control trancode usage by assigning a security-category name to a trancode or group of trancodes. You can then group security categories into lists to apply to these three entities.

You can restrict the trancodes a program can access on output, as well as the trancodes that can be entered from or received by a station.

See the *A Series Communications Management System (COMS) Programming Guide* for more information on using security categories at the application-program level.

Security and Monitor Messages

You can use COMS to track certain kinds of system messages. You implement message tracking by creating a monitor-station list named MONITOR. You can then issue the *MONITOR* COMS command from a command station. Refer to the MONITOR command description in the *COMS Operations Guide* for information on the options available to you.

If you have InfoGuard security enhancements, COMS enables you to create a security-station list named SECURITY, to track the operating system security messages. See the *A Series Security Administration Guide* for details of InfoGuard security.

Note that a station can be in both the monitor-station list and the security-station list without receiving duplicate messages. When a user with the security classification of SECURITYMSGUSER signs on at a security station, COMS security and log-on/log-off messages are sent to the security station instead of to each monitor station. You can distinguish monitor messages from system security messages by their format. Monitor messages are displayed with the prefix

```
MONITOR >>>
```

Security messages are displayed with the prefix

```
SECURITY >>>
```

If no monitor stations are defined, the security station receives both monitor and security messages. If there is no monitor-station list and no security-station list, exception errors are reported at the operator display terminal (ODT).

Only someone who is designated as a COMS control-capable user can create or change the security station list.

Note: *After a station is assigned continuous log-on capability, that station is logged on automatically even if it is a security station. To ensure that the SECURITYMSGUSER usercode is not misused, do not assign continuous log-on capability to SECURITYMSGUSER usercodes.*

See the *Security Administration Guide* for more details about the SECURITYMSGUSER usercode.

Other Forms of Security with COMS

You can specify other forms of security at the application-program level. Because COMS passes security-related information to application programs on input, you can use this information programmatically.

You can find more information on creating and using COMS windows in the discussion of COMS in Section 10 of this guide. For a complete discussion of the COMS Utility, including information on COMS windows and security features, refer to the *A Series Communications Management System (COMS) Configuration Guide.*

Maintaining Accountability through Auditing

Auditing events and actions on the system permits the detection of break-in attempts, resource misuse, and other security-relevant activities.

Auditing is based on selective examination of events and system actions. A file called SYSTEM/SUMLOG stores a record of activities, such as session log-on and log-off, the beginning and ending of tasks, attempts to log on with an incorrect usercode, and unauthorized attempts to change or copy a file. A detailed description of the SUMLOG utility and the SYSTEM/SUMLOG file is presented in the *System Software Support Reference Manual.*

Several tools, primary among them a utility named LOGANALYZER, are available to enable the security administrator to look at those entries in the SYSTEM/SUMLOG that are of interest.

Access to the system log is always controlled by the System Data Access (SDA) support library. This library is privileged on an InfoGuard system and so has complete access to the system log. The SDA support library filters the SUMLOG and allows access to certain log records, depending on the usercode status being used. On systems where the system log is private, the SUMLOG records available to end users are limited to the following:

- Maintenance records
- Halt/load records
- All log records produced by actions that user took on the system, except for actions that were security violations (such as an attempt to log on with an incorrect usercode/password combination)

When the system log is public, the SDA support library does not filter the SUMLOG, and any user has access to all log records.

When undesirable activity is detected by an audit, the security administrator can take steps to remedy the problem. Such steps might involve changes in the system security policy, changes of passwords that might have been compromised, intensive auditing of the actions of particular users, changes in the system access rights allowed certain users, and, in extreme cases, denial of physical access to the system for some users.

Section 4
Allocating System Files

System performance depends upon the efficient operation of certain critical system files. The best system performance is achieved if these files are spread over different disk families in such a way that attempts to access the files do not interfere with each other.

This section provides details about how often the various system files are accessed, indicates the consequences of a failed disk containing these files, and provides guidelines for allocating individual system files to disk. The following topics are covered:

- System startup
- Characteristics and requirements of the various system files
- Allocation of system files to improve system performance and to ease recovery from certain errors

System Startup

Most Unisys A Series systems are delivered to the customer preinitialized, meaning that the MCP is already resident on disk. Some systems, however, require the use of the LOADER utility to initialize the system for the first time. The functions of LOADER include setting up the halt/load family and copying the MCP to the halt/load family.

The next step is to load the remaining system files. The easiest way to do so is to use the Simple Installation (SI) program (also known as SIMPLEINSTALL) that Unisys provides for this purpose. Refer to the installation guide for your system for more details.

Once your system is running (that is, the system software is installed), you can begin to move system files to the most appropriate location. There are several ways you can go about doing so. Here are some considerations:

- If you used SIMPLEINSTALL to load files, you should continue to use that method for moving files because the Simple Installation (SI) program keeps track of the location of the system software files. If these files are moved by another method, then the Simple Installation (SI) program will not have a record of their location.
- If you used the WFL *COPY* statement to load files, you can continue to use this statement as well as system commands such as DL (Disk Location), SB (Substitute Backup), SI (System Intrinsics), and SL (Support Library).

After the system is running satisfactorily, you should create one or more alternate halt/load families if you have the disk space. Procedures for this are given under "Making Alternate Halt/Load Families" in Section 5.

System File Characteristics

All the system software code files are stored as disk files. The following pages describe these files, which include

- MCP code file
- System library code files
- System intrinsic code files
- Overlay files
- Job description files
- Job files
- Printer backup files
- System log file
- Disk access structure file
- Usercode description file
- Sort files

The way these system files are dispersed among disk families greatly influences how efficiently the system operates.

MCP Code File

The current MCP code file (the one the system is running on) is accessed frequently, and all accesses are time-critical. Most accesses of the MCP code file are caused by presence-bit interrupts, which in this case notify the MCP that an MCP code file segment is needed that is not in main memory. The system then reads the code file segment into main memory from disk. The MCP cannot predict which code file segments are going to be needed to be read into main memory, so it cannot read them into a look-ahead buffer ahead of time.

The disk unit where the MCP code file is stored is called the halt/load unit or the boot unit. Corruption of the MCP code file might require a halt/load. You restore the file by recopying the original file and then using the CM (Change MCP) system command to designate the new code file as the halt/load MCP.

The WM (What MCP) system command can be used to determine which MCP code file the system is running on.

System Library and Intrinsic Code Files

The system library and intrinsic code files are files such as the following:

- System libraries that are linked to function names established through the SL (Support Library) system command
- System command and system intrinsics established with the SI (System Intrinsics) system command

The characteristics of these files are similar to those of the MCP code file. Failure of the disk containing these files usually does not require a halt/load; however, a failure can cause jobs using these files to be discontinued.

Other System Code Files

Other system files, such as language compilers, have characteristics similar to those of the MCP code file. Their failure usually does not require a halt/load, but can cause jobs using them to be discontinued. Unisys suggests that these files be grouped onto a single family so that they can be conveniently referred to through family substitution. Family substitution is discussed in Section 5, "Understanding the Disk System."

Overlay Files

Overlay files are used to store data on disk when data is overlaid in main memory. These files do not have headers in the flat directory (the part of the disk that stores information about the files on that disk). Overlay files are accessed frequently, especially if main memory is full or if the overlay goal (the OLAYGOAL memory management factor) has a value greater than 0. All overlay file I/O operations are time-critical, and if two or more programs try simultaneously to access the family where these files are stored, system performance can be seriously degraded.

Failure of the family where overlay files are stored can cause jobs to be discontinued and can require a system halt/load. It is neither necessary nor possible to save or restore the contents of overlay files.

Job Description Files and Job Files

The job description file contains information about job queues and job files. The job description file is named *JOBDESC and can be stored on any disk family. Only one job description file can be in use at one time.

You can display the location of the job description file by using the DL JOBS system command. By default, the job description file is stored on the halt/load family. You can move the file to another family either through the Simple Installation (SI) program or through the DL system command.

Job files are code files that are compiled by the WFL compiler. Job files do not have headers in the flat directory. These files are accessed regularly, and all JOBDESC and job file I/O operations are time-critical. Failure of these files can cause a halt/load.

Failure of the JOBDESC file results in the loss of the input job queues, information about the queues, and job log output for jobs that are waiting to be printed. The JOBDESC file also contains certain system operational characteristics such as ADM (automatic display mode) command settings and TERM (Terminal) command settings. This information is lost if the JOBDESC file fails or is moved to another family.

The contents of the job description and job files cannot be recovered, but they can be re-created by rerunning the jobs. If persistent I/O errors occur for the JOBDESC file or the job file, you should use one of two alternatives:

- Enter *??RJ* (the Remove JOBDESC File primitive system command) and then halt/load the system to build a new JOBDESC file.
- Use the DL (Disk Location) system command to change the location of the JOBDESC file to another family. Note that the DL system command does not move the contents of the current JOBDESC file to another family.

Building a New Job Description File

The ??RJ primitive system command can be used to prevent the current job description file from ever being reused. The effects of the command vary depending on whether the DL command is also used to change the DL JOBS ON < family> setting. In the following descriptions, the job family and job description file in use before the halt/load are referred to as *old,* and those in use after the halt/load are referred to as *new:*

- If the ??RJ command is used, but the setting of *DL JOBS ON <family>* is not changed, then a new job description file is created on the old job family when the next halt/load occurs. The title of the old job description file is changed to *OLDJOBDESC and the FILEKIND file attribute is changed from JOBDESCFILE to DATA. The old job description file can never be reused as a job description file, but it remains resident until removed. All jobs, job queue definitions, and other settings stored in the old job description file are lost at the time of the halt/load.
- If the ??RJ command is used and the setting of the *DL JOBS ON <family>* command is changed, then the system searches for a valid job description file on the new job family when the next halt/load occurs. If such a file is present, the system uses it; otherwise the system creates a new job description file on the new job family. Meanwhile, the title of the old job description file remains unchanged, but the FILEKIND file attribute changes to LIBRARYCODE. The old job description file can never be reused as a job description file, but it remains resident until removed. All jobs, job queue definitions, and other settings stored in the job description file are lost at the time of the halt/load.

  If a *DL JOBS ON <family>* command later specifies the old family and the old job description file is still resident on that family, then the title of the old job description file changes to *OLDJOBDESC and the FILEKIND changes from LIBRARYCODE to DATA after the halt/load.

You can display information about job description files by using the PD (Print Directory) system command, the MARC *FILES* command, or the CANDE *FILES* and *LFILES* commands.

Relocating the Job Description File

The *DL JOBS ON <family>* system command specifies the family on which the job description file will be created after the next halt/load. This family is referred to as the *job family.* This command has the following effects:

- The current job queue definitions are not copied into the new job description file. Consequently, you must use the ML (Mix Limit), MQ (Make or Modify Queue), DQ (Default Queue), and UQ (Unit Queue) system commands to re-create the desired job queue structures.
- The jobs that were waiting in job queues at the time of the halt/load are not requeued. Also, the jobs that were executing at the time of the halt/load are not restarted. The users must resubmit these jobs.
- Settings created by certain system commands are stored in the job description file. These settings are not retained, so you must restore them after the halt/load. (Note that this is good reason for establishing a supervisor program.) The following system commands are affected: ADM (automatic display mode), PA (Peripheral Association), and TERM (Terminal).
- If the DL JOBS system command specifies a job family where an old job description file still resides, then after the halt/load the CONTROLLER uses the old job description file on the new job family rather than creating a new job description file. (The CONTROLLER is the MCP independent runner that handles the job queue mechanism and most system commands.) All the job queue definitions and other settings in the old job description file on the new job family are used again. Any jobs that were waiting in the old job queues are restored to those queues and eventually executed. Any jobs that were executing when this job description file was last in use are restarted.
- The old job description file remains present on the original job family after the halt/load. However, this file is no longer used. You can remove it by using a CANDE *REMOVE* command or a WFL *REMOVE* statement. Note that the system does not allow the current in-use job description file to be removed.

You cannot preserve the old job queue definitions and jobs by copying the job description file to the new job family before the halt/load. In fact, the system does not allow the job description file to be copied to another family because the job description file includes directory information for the job files. This directory information would not apply to a different family.

Job Description Files on Multiprocessor Systems

If you use the RECONFIGURE (Reconfigure System) system command to partition a system into separate groups, each group has its own job description file. The location of the job description file for each group is specified in the system configuration file. For information about specifying this information in the system configuration file, refer to the discussions of the DL system command in the *A Series System Configuration Guide.*

Some jobs might not restart after the reconfiguration if the reconfiguration reduces the number of job description files in use or causes different job description files to be

used. For example, consider a system with two one-processor groups, each of which has a job description file. When they are joined into a two-processor group, only one job description file can be used. Only the jobs and job queues recorded in the selected job description file are restored. If the job description file is from one of the one-processor groups, the jobs from that group restart. The jobs described in the other job description file do not restart at this time.

If the same system is reconfigured to restore the two one-processor groups, then each group has its own job description file again and restarts any jobs that are described in that job description file. If one of the groups is assigned to use the same job description file as that of the two-processor group, all the jobs that were running on the two-processor group are restarted in the assigned group.

Printer and Punch Backup Files

Backup files are accessed frequently, but I/O operations are buffered and thus are not time-critical. If these files are placed on the same disk with other system files, however, the I/O operations of printer and punch backup files can interfere with the time-critical I/O operations of other system files. The I/O operations of the job description file and overlay files are particularly time-critical.

Failure of a printer or punch backup file can cause the loss of some output or cause some jobs to be discontinued. Data in these files cannot be recovered, but the jobs can be rerun to produce output again.

System Log File

The system log file records the activities performed on the system. This file is titled SYSTEM/SUMLOG and is accessed regularly. Its failure usually does not require a halt/load.

The data in the log cannot be recovered after a failure, which can pose a problem if your installation uses the log to provide information for billing purposes. If you use the log for billing, then you need to make backup copies of the log. If there are persistent errors in the log file, you can use the TL (Transfer Log) system command to save the current log file and create a new one.

Disk Access Structure File

The disk access structure file is a regularly referenced file whose failure can cause jobs to fail, can cause a family or the catalog to be rebuilt, or can require a halt/load. On cataloging systems, this file is named SYSTEM/CATALOG/<family index number>. On systems that are not using the cataloging feature, this file is named SYSTEM/ACCESS/<family index number>.

Restoration of the disk access structure file generally is not necessary on noncataloging systems that are not using the tape security subsystem. If you re-create a disk access structure by rebuilding a family, no data is lost. If you are using the tape security subsystem, restoration is necessary on noncataloging systems. (You can activate the

tape security subsystem by using the SECOPT TAPECHECK = AUTOMATIC system command.) You can write a program to make backup copies of the volume directory by using the GETSTATUS and SETSTATUS calls in DCALGOL. For more information about GETSTATUS and SETSTATUS calls, refer to the *GETSTATUS/SETSTATUS Reference Manual.*

On a cataloging system, restoration of the disk access structure file is difficult and usually involves the loss of some catalog backup information. Refer to the discussion of "Duplicating the Catalog File" in Section 5, "Understanding the Disk System."

Usercode Description File

The usercode description file, known to the system as SYSTEM/USERDATAFILE and usually referred to as the USERDATAFILE, is generally a small file that contains usercodes and the attributes allowed for each usercode for acknowledged users of the system. Most I/O operations involving this file are not time-critical. A copy of this file should be kept for backup although you should take security precautions with the copy because of the sensitive nature of the information it contains.

For more information about the usercode description file, refer to the *Security Administration Guide.*

Sort Files

Sort files are temporary files used by the sort intrinsic, a function contained in the MCP. The I/O operations of sort files do not affect the performance of the MCP, but do affect the performance of programs that use the sort intrinsic. Also, if the sort files are on the same family with other system files, sorting can interfere with the time-critical I/O operations of these system files. Sorting uses large temporary files, and it is not necessary to save backup copies.

System File Allocation

System performance and the ease of recovery from catastrophic errors primarily depends on your choice of families on which to place various system files. The following are some decisions you can make that affect performance and recovery:

- Location of the MCP code file (and thus what the halt/load family is)
- Location of the system intrinsics and the system library files
- Location of the following system files:
  - Printer backup files
  - Job description files
  - System log files
  - System access structure file

  The locations of these files are specified with the DL (Disk Location) system command.

You can mix some or all system files with files used by application programs, or segregate system files from files used by application programs. These decisions involve tradeoffs.

For maximum system performance it would be ideal to store every system file on a different disk and never put any other files on those disks. But doing so is not practical for two reasons: some of the system files are very small and a large amount of disk space would be wasted; the system would be vulnerable to the failure of any of several disks or disk drives. When deciding where to locate disk files, consider the following criteria:

- Amount of storage space needed
- System performance (speed)
- The probability that the failure of a single disk will cause a service interruption (such as DISK DRIVE NOT READY errors requiring a halt/load)
- Ease of recovery from damaged files or media

If your installation has a limited number of disk families, you might have to group all the system files described previously on the halt/load family. This family might have to be named DISK if you run programs that require this naming convention. If your installation has several disk families, however, you can improve performance by distributing the system files to different families.

With a limited number of disk families, your choices for grouping system files should take into account the impact on system performance. One arrangement could be to store the disk access structure, the USERDATAFILE, and the overlay files on the halt/load family. Application files such as seldom-changed code files also could be grouped with the system files.

The failure of a system file or the family it resides on usually does not result in the permanent loss of any important data unless the file is the USERDATAFILE, the system log, or the catalog at an installation that uses the cataloging feature and/or the tape security subsystem. Creating a new file or recopying the original file usually solves the problem caused by the failure of system files other than the three files previously mentioned. Sometimes you must move a system file to another family until the original family is restored. Use the DL system command to move a system file from one family to another.

Most system files do not change in ways that require saving frequent backup copies. Unisys recommends that system files be segregated from application data files that do need frequent safety backups. This way, total failure of a family containing system files does not entail the lengthy recovery operations that might be required for application data files. Similarly, total failure of a family dedicated to application data files does not harm the operations of the operating system, and recovery actions to restore the application files can take place while the operating system is fully operational. Thus, it may be simpler (and preferable) to place static system files and static application files on one family and dynamic application files on a different family.

System performance usually improves if you segregate the halt/load family and the overlay files from all other files. MCP code file presence-bit handling and overlay handling can significantly degrade system performance. Consequently, the worst possible

performance is produced if these time-critical files share families with application programs that perform many I/O operations.

Unisys suggests that the halt/load family be a single disk family with no application program files stored on it. The halt/load family should not be a multidisk family unless you want to have duplicate MCP code files, which are explained under "Duplicating the MCP Code File" in Section 5.

System performance usually improves if the halt/load family is reserved for the files that must be there: the MCP code file and the SYSTEM/TRAINTABLES file, which controls printer operation. Systems that use network support processors (NSPs) must have the file FIRMWARE/NSP stored on the halt/load family, and systems that use line support processors (LSPs) must have the file FIRMWARE/LSP stored on the halt/load family. On A 1 through A 6 systems, Unisys suggests that you store the BOOTCODE file on the halt/load family.

Section 5
Understanding the Disk System

The disk system on Unisys A Series systems consists of the following:

- The disk media
- The disk drives, usually referred to as disk units
- The disk drive controller, which controls the disk units and transfers information between the host system and the disk units
- The I/O controller, which provides the interface between the host system and the disk drive controller
- The software that controls the structure and operation of the disk subsystem and provides the structure for the information stored on the subsystem

Types of Disks Used on A Series Systems

Unisys A Series system software treats various disk models as logically identical. Disks are logically organized in one of two ways to be used on Unisys A Series systems: as native-mode disks or interchange disk units. However, interchange disks are seldom used, so this guide is concerned only with native-mode disks.

When you use the RC command to reconfigure a disk, the default is for native-mode disk. You do, however, have the option of designating the disk as an interchange disk.

Preparing a Disk for Use

Before a disk can be used for the first time, be sure that the following procedures have been performed:

- The initialization, verification, relocation (IVR) procedure

  Before Unisys ships a disk to a customer, it uses the IVR procedure to write sector boundaries and the label on the disk. Your field engineer or customer service engineer can verify that the IVR operation has been performed.

Caution

The IVR procedure should be used only under the direction of your Unisys field engineer.

- The RC (Reconfigure Disk) system command

  Before placing a disk into active operation, use the RC command to prepare the disk. With this command, you specify the family name and serial number to assign to the disk. This information is stored in the disk label. The OL (Display Labels and Paths) system command returns disk label information.

Disks are distinguished from one another when they are online by serial numbers, which are used to give each disk a unique, permanent identification to aid in keeping track of disk use, as well as logging errors or other problems. Some choices for a serial number are

- The number listed on the bottom of the disk by the manufacturer
- A reference number such as the date that the disk was first used
- For nonremovable disks, a serial number the same as the unit number of the disk drive on which the disk is mounted

Each disk unit that is online to the system must have a unique serial number. Serial numbers are 6 digits long and should be assigned in a systematic fashion.

When you use the RC command, you can assign a name to the OWNER clause. The OWNER clause provides a safety feature to ensure that you actually want to reconfigure the disk. If another RC command is attempted for that disk, the system displays a message on the ODT specifying the value of the OWNER clause and asks the system operator if the reconfiguration is permissible. When you reconfigure or purge a disk, any files on that disk are made permanently inaccessible. Care should be taken so that a disk is not reconfigured by accident. As an added safety precaution, the system requires the old name of a disk before the reconfigure operation is performed.

Disk Families

A disk *family* is a disk or group of disks with a common name that are treated as a single, logical entity by the system. By this definition, a disk family can consist of one disk unit or several disk units. Disk families are created with the RC command.

The field engineer assigns a unique unit number to your disk units when they are installed. To find out what the disk unit numbers are, use the PER PK command.

When you create a family with the RC command, the first disk unit you assign to that family is called the *base unit.* This unit normally contains the *flat directory,* which is a special structure on each family that the system uses to locate files on that family. You can create duplicate copies of the flat directory on continuation units of a multidisk family.

The family name and other information about a disk is stored in a structure called the *label.* The label also contains the serial number assigned to the disk, and a pointer to the flat directory of the disk, if there is a flat directory. When a program needs to access or allocate a file, the system uses the family name to locate the disk that contains the file.

Disk units should be assigned unique family names unless they are to become part of the same family. If you try to use a disk that has the same family name as that of another disk on the system and the two disks are not in the same family, the system issues an error message.

The index number of a disk unit identifies its relative assignment to the family. The initial base unit of a family is assigned family index 1. Multidisk families will have continuation units that are assigned family index numbers incremented by 1 to a maximum of 255. Note that there is no connection between the serial number of a disk unit and its family index number.

Programs can use the FAMILYINDEX file attribute to distribute files or areas of files to particular disks in the family, otherwise the MCP will automatically handle allocations.

The file name of the flat directory is SYSTEMDIRECTORY/<nnn>, where <nnn> is a 3-digit number and is the family index number of the disk on which the flat directory is stored.

The PER PK system command displays the family name, serial number, and family index number of all the online disks on the system.

Multidisk Families

Large databases or large production or user groups may require more disk space than is contained in one disk drive. For these reasons, families can consist of more than one disk unit. Such families are called multidisk families. In multidisk families, files are spread out across all members of the family.

Additional disks can be added to the family as continuation units, and the system logically links the family members together so that they are treated as a single entity. A family can have up to 254 continuation units. Continuation units can contain a copy of the flat directory file.

When the system needs to access a multidisk family, the base unit and any continuation unit that contains a copy of the flat directory must be online. If the file to be accessed is spread over several family members, continuation units must be brought online when areas are needed that are stored on those continuation units. If a continuation unit must be brought online, a message is sent to the ODT. Refer to the RY, UR, and ACQUIRE system commands in the *System Commands Reference Manual* for information about bringing disk units online.

The system can allocate different areas of a particular file on different members of the same family. You have the option of designating which family members are to receive areas of the file by using the FAMILYINDEX file attribute. The system cannot spread a file over disks that are in different families.

To create a multidisk family, use the BP (base pack) option of the RC system command. The base pack option logically connects the disks. Keep in mind that the RC system command makes any files that previously were on a disk inaccessible. Adding a continuation unit to an existing family does not harm the files already on the family, but it will erase any files that are on the new continuation unit.

Family Substitution

You might find it convenient to group the disk resources of your installation so that each department or other organizational group using the system has access to its own disk family where its private files are stored. These organizational groups can still share the disk family on which the system compilers and other system files are stored.

Family substitution helps you establish a means of organizing disk family access according to the needs of the users on your system. Family substitution enables you to designate which private family a particular program, job, or usercode should access for private files and which system family it should access for system files.

You can assign a default family specification in the following ways:

- With the MAKEUSER utility for usercodes

  The MAKEUSER utility can establish a default family specification for each usercode so that the system is directed to first look for files on a departmental family.
- With the CANDE or MARC *FAMILY* command for sessions
- With task attributes within programs
- In jobs with a WFL statement or the MQ (Make or Modify Queue) system command

Unisys A Series system software and many application programs are written so that the system by default searches for a specified file on a family named DISK if no other family name is specified. (Certain system programs expect certain files to be on a family named DISK.) Generally, placing all files (both system and application) on the DISK family is inefficient and can cause performance problems, so family substitution can be used to redirect files to other families. The family specification can designate both a substitute family name and an alternate family name.

To use family substitution to distribute files to families other than DISK, use the family specification statement with DISK as the target family name, a department's private family as the substitute family name, and the general family that contains system code files as the alternate family name.

For example, for personnel in the accounting department, you might set up the usercodes with a family specification that accesses application programs and private files on a family named ACCTS and general system files on a family named SYSTEM. The CANDE command to do this would be

```
FAMILY DISK = ACCTS OTHERWISE SYSTEM
```

When a program needs to access a file that is specified as being on the target family (can be DISK or any other family name), the family specification causes the system to search on the substitute family and then the alternate family if the file is not found on the substitute family.

Once a family specification statement is in place, all new files that are specified as being directed to the family named DISK are stored on the substitute family instead.

On a system with several disk families, Unisys suggests that the system compilers, code files, and general support programs be stored on an alternate family that is shared by all the departments. On a small system that does not have a large number of disk families, all system files could be grouped on the halt/load family. This family could be called DISK or anything else you want.

Using family substitution does not prevent a program or user from accessing or allocating a file on a particular family other than the substitute family or the alternate family. If a family name is used that is not the target family in the family specification statement, the system places the file on the specified family. Remember that the *ON <family name>* clause can be used by a program run or file access to locate a particular file.

For more information on the family specification statement, refer to the *A Series Work Flow Language (WFL) Programming Reference Manual* and the discussion of usercodes in the *Security Administration Guide.*

Accessing Disk Files

Computer performance depends to a large degree upon the ability to perform efficient I/O operations. Unisys systems use an access method consisting of two parts: the disk access structure used by the entire system, and the flat directory on each disk family.

The system has a special disk file called the disk access structure file or the catalog file. This disk file is stored on one family in the system. You can designate the family to be used with the DL CATALOG system command. The catalog disk file contains several components, which include the pack access structure table (PAST) and the file access structure table (FAST). On tape security subsystems, the catalog file also contains the volume directory. On systems that use cataloging, the catalog file also contains the catalog and the volume library.

The disk access structure functions differently on cataloging and noncataloging systems. The disk access structure for a cataloging system contains entries for all available versions of the file. On a noncataloging system, the disk access structure contains only entries for the files that are currently online on the system.

The file name of the access structure on noncataloging systems is

```
SYSTEM/ACCESS/<family index number>
```

The file name of the access structure on cataloging systems is

```
SYSTEM/CATALOG/<family index number>
```

The family index number indicates which member of the catalog family contains the access structure.

The operating system constructs the access structure the first time the system is initialized so that the access structure contains entries for each family that is online at that time. Each time a family is brought online, entries for the files on that family are entered into the access structure. When a family is removed with the CLOSE (Close

Pack) or POWER (Power Up/Down) system command, references to its files are removed from the access structure unless it is a cataloging system and the family is volumed.

Volume Directory

The volume directory is a data structure in the tape security subsystem that stores information about tape volumes by serial number at your installation. The volume directory includes the following information for each tape volume family:

- Current volume name
- Tape volume ownership
- Tape file security attributes

The volume directory is used by the system when a tape volume is readied or purged and whenever a tape file is opened for input or output. The tape volume directory is a feature that is available only with InfoGuard security-enhancement software.

If you designate the SECOPT TAPECHECK option equal to NONE, then the system does not create, refer to, or update the volume directory data structure. To have volume directory capability, you must designate the SECOPT TAPECHECK option equal to AUTOMATIC.

Note that a systems programmer can use the GETSTATUS and SETSTATUS intrinsic routines in DCALGOL to save and restore the volume directory. GETSTATUS is used to make a copy of all the data records in the volume directory, while SETSTATUS calls are used to delete serial numbers from the volume directory and add records to it. For more information on GETSTATUS and SETSTATUS, refer to the *GETSTATUS/SETSTATUS Reference Manual.*

Rebuilding Families

The only time you will be aware of the access structure is during the process known as family rebuilding. When a family is rebuilt, the system constructs or reconstructs the FAST entries for a family by sequentially reading its flat directory.

Families are rebuilt in the following situations:

- Automatically when you bring a base unit online on a noncataloging system and no local access structure table (LAST) is found on the base unit
- Automatically to recover from certain directory errors when they occur
- When you use the RB (Rebuild Access) system command to initiate a rebuild

If while a family is being rebuilt, two disk file headers are found with the same file name, the rebuild process bypasses the second file. If no operator action is taken, the second copy of the file cannot be accessed. However, if the first copy of the file is removed or renamed (using the REMOVE or CHANGE system commands), the next rebuild of the family will reveal the second copy of the file.

Local Access Structure Table (LAST)

To eliminate the need to rebuild families on a noncataloging system each time a disk unit is brought online, the system uses the LAST to restore the FAST, instead of rebuilding the FAST from scratch. The LAST is written to the base unit whenever the base unit is closed using the CLOSE system command, freed using the FREE system command, or powered off using the PO system command. When the disk unit is readied, the system attempts to restore the FAST from the LAST. If the restoration is successful, the family is not rebuilt.

However, the LAST is not created under the following conditions:

- The system is a cataloging system.
- The disk unit is not write-enabled when closed.
- The disk unit has no room for the LAST when the unit is closed.
- You turn off the disk unit manually without first entering a CLOSE, FREE, or PO system command.
- I/O errors are encountered during the creation of the LAST.
- The disk unit is mirrored.

Managing Disk Space

Disk space is a critical commodity on a computer system, and the inefficient use of disk space can have drastic effects on system performance. It is important to decide how often to investigate and analyze disk usage for all disk families and to implement policies and procedures for performing disk cleanups. To obtain information about how disk space is being utilized, you can use the DU (Disk Utilization) system command, the *PDIR* (Process Directory) MARC command, and the FILEDATA utility.

By using the DU system command, you can get information about space utilization for any disk family or individual disk unit of a multidisk family. The DU command displays the following information:

- The number of available sectors on the disk
- The number of sectors on the disk that are in areas that are smaller or larger than 504 sectors
- The number of disk rows that are 504 sectors long
- The size (in sectors) of the largest available area on the disk

The DU command will also report the following information when used to check the WORM media:

- The number of sectors in use
- The number of sectors available
- The total capacity of the media

The *PDIR* MARC command enables you to perform any of the following file management functions:

- Display a defined set of file attributes for each file in a list.
- Select one or more files that meet a user-defined set of criteria, including wild-card searching on file names.
- Sort a list of files according to the values of one or more file attributes.

When you enter the PDIR command alone, the system displays all files under the current usercode on the current disk family.

You can also enter the PDIR command with a string composed of the names of one or more files for which you want to display information. The string can be the name of an individual file or directory, optionally including a usercode and a family name. The string item SHOW can be used with the PDIR command to display selected information about the specified file or files.

The PDIR command is unlike other MARC commands in that it completes the searching process before displaying the requested information. As a result, PDIR commands can be very time-consuming.

As a safeguard against excessive consumption of system resources by PDIR commands, MARC assesses each PDIR command it receives to determine whether its processing would consume excessive system resources. If so, MARC rejects the command and issues a message suggesting that you reenter the command at a later time.

Note: You cannot execute the PDIR command if only one copy of MARC is running. The number of running copies is determined by COMS and always falls between preset maximum and minimum numbers. You can use the Program entity of the COMS Utility program to display the values of these numbers and to modify those values. The default maximum number is 2.

For more information on the *PDIR* MARC command, refer to the *A Series Menu-Assisted Resource Control (MARC) Operations Guide.*

The FILEDATA utility generates two reports that are useful for analyzing disk usage. The names of these reports, their characteristics, and their possible uses are presented in Table 5–1.

Table 5–1. FILEDATA Disk Usage Reports

Report	Characteristics	Uses
CHECKERBOARD	Shows the space on the specified disk family that is in use and the space that is available.	Use this report to determine whether you need to consolidate space on a disk. If the report shows a large number of small areas of free disk space, then the disk space is checkerboarded and you should attempt to consolidate the disk.
STRUCTUREMAP	Shows all the areas of a file on any disk family and where those areas are located.	Use this report to determine whether a part of a known file exists on a particular family index and to determine the extent of fragmentation of files that are stored on multidisk families.

If you are having problems with disks becoming checkerboarded, there are two methods you can use to solve the problem: you can consolidate the disk space by using the SQUASH command, or you can repack the disk.

Consolidating Disk Space with the SQUASH Command

The SQUASH command causes the MCP to attempt to consolidate the available areas on a disk. If used consistently, this is the most nondisruptive way to consolidate disk space. However, the results of a SQUASH command depend on the following factors:

- Files in use. The larger the number and size of areas owned by in-use files, the less effective will be the results of a SQUASH command.
- Largest available areas. A SQUASH command cannot relocate data in files whose AREASIZE is larger than the current largest area. If a disk family has many files whose AREASIZE is larger than the largest available area, it will be difficult to consolidate the disk space with the squash operation.
- Severity of fragmentation. If there are more available areas on the disk whose size is smaller than the typical file AREASIZE, the more likely it is that a squash operation will not be effective.

Because the effectiveness of a squash operation is so dependent on a number of factors, Unisys recommends that you take the following long-term measures:

- Schedule periodic (from nightly to weekly depending on amount of activity) squash operations on all families. Perform the operations at times when the fewest files are open (none is best) and the most space is available.
- Adjust the area sizes of the disk files as well as the family assignments for various types of files. These measures are particularly useful if large files are frequently created with the same name as existing files (which causes the existing file to be removed when the new one is created).

There are some specific measures that you should take if a squash operation fails to increase the available space on a disk. Simply repeating the SQUASH command a number of times might solve the problem, because each squash operation can affect a different area of the disk family.

If no significant improvement occurs after 10 to 12 successive SQUASH commands, do any of the following that might be appropriate, and then repeat the SQUASH command:

- Remove all files that can be discarded, and any files that could be restored from a backup copy of the disk. Files with large area sizes are the most useful ones to remove.
- Close as many files as possible that are currently open on the disk family. If a job or task is waiting because of a SECTORS REQUIRED message, discontinue the job or task if it can easily be rerun later. However, do not terminate a database that is waiting for sectors unless there is no alternative.

- If the OLAYSCOUT independent runner is waiting because of a SECTORS REQUIRED message, use the DL OVERLAY system command to move overlay areas to a different family. The move need only be a temporary measure, though you can leave it as a permanent change if it improves performance. Note that the command does not move overlay areas of tasks that are currently active. However these areas are released as the tasks terminate, and new tasks use the new family for overlays.

Repacking a Disk

Repacking a disk is a much more drastic measure than performing a squash operation and should seldom be necessary. However, if using the squash operation is not succeeding in correcting a checkerboarding problem for a disk, then you may want to consider repacking the disk family (or one unit of a multidisk family).

To repack a disk family, there must be no one using the family you need to repack and you must have an available disk onto which you can move the files from the disk you want to repack. Use the following procedure:

1. Use the LB (Relabel Pack) system command to rename the disk family that you want to repack so that no one can access the family during the repacking operation.
2. Use the RC (Reconfigure Disk) system command to name a new family to receive a copy of the files from the family that needs to be repacked.
3. Use the COPY command to transfer files from the family to be repacked to the new family you just created.
4. Once all files have been copied to the new family, use the LB command to rename the new disk (the one to which you just transferred the files) with the original family name.
5. Use the RC command to designate the disk from which the files were transferred as a scratch unit.

The RC command makes one contiguous area on the disk beginning at a low address just above the directory. When the COPY command is used to transfer the files, all the space for each file is allocated at one time; new space is always taken from the lowest area in the free space pool large enough to contain the file. The result of this process is a well-packed disk with all the files located at the low address areas and all the available space in one large group at the high address areas of the disk.

If you need to repack a multidisk family, you must create the same number of continuation units on the destination family before you begin copying files. Because of the way the COPY command works, areas of the files being copied are distributed evenly across all family members. Thus, the available space is evenly distributed among the different family members.

Another good reason for avoiding the use of large multidisk families is that if you need to repack a family composed of four disk drives, you must have four available (scratch) disk units to copy the files onto.

Note: *One other alternative you have for rectifying problems with disk space allocation is to copy the problem disk units to tape (with the COMPARE option) and then reload the files from tape onto the original disk units.*

However, this method can take a considerable amount of time and there is a danger that some files can be lost because of tape parity errors.

Also, when considering how you want to manage your disks, remember that the REPLACE system command does not perform the same function a the COPY command. The REPLACE command performs a bit image copy of the disk; thus, the new disk generated by this operation is an exact replica of the original disk, complete with checkerboarding still intact.

Safety Mechanisms

The MCP code file, the flat directory of each family, and the catalog file on cataloging systems are vital for proper system operation. Although the failure of one of these critical system files is rare, it is still important to ensure that a problem with one of these files does not cause a service interruption. These three system files can be duplicated so that the system can still operate after the failure of a disk on which a critical system file is stored.

Comparative approaches to duplicating these files and an explanation of how to make alternate halt/load families are presented.

The most basic safety mechanism is to make backup copies of user disk files on tape or other disk families. Each installation's requirements vary depending on how often the files are changed, how critical the information on each file is, and whether the information can be more easily restored through an audit or other means.

The emphasis of this discussion is on safety mechanisms for system files rather than safety mechanisms for user files. The FILEDATA utility can produce a list of when each user file was last changed or updated. The FILECOPY utility can automatically create WFL jobs that make backup copies of user files.

Refer to the *A Series System Software Utilities Operations Reference Manual* for more information about the FILECOPY and FILEDATA utilities. For information about making backup copies of database files, refer to the *DMSII Utilities Operations Guide*. You can also refer to the *File Attributes Reference Manual* for information about the DUPLICATED and COPIES file attributes.

Duplicating the Flat Directory

As the number of disks in a family grows, a large amount of data becomes vulnerable to a single failure of the base unit. If the entire base unit is unusable, none of the data on any of the disks in the family can be accessed. If a particular record in the flat directory is corrupted, the file referenced by that record is not available. The DD (Directory

Duplicate) system command can be used to avoid these problems by making an exact duplicate of the flat directory on another member of the family.

The duplicate flat directory is titled SYSTEMDIRECTORY/<family index number>. Up to three disks in each family can have a copy of the flat directory (the base unit and two continuation units). When the system halt/loads or the family is readied, any of the family members containing flat directories can be used as the base unit. The system arbitrarily chooses one as the base unit and treats the others as continuation units with duplicate directories.

All the family members that contain a copy of the flat directory must be online when you access that family, so that all changes in the directory can be made simultaneously to all copies. If some of the disks containing directories are not mounted, then the missing ones become outdated and their flat directories are not accepted by the system as up-to-date duplicates.

If a disk with an outdated directory is subsequently mounted, you must be careful to avoid designating the outdated disk as having the most up-to-date directory. If the outdated disk is mounted when the up-to-date base unit is online, the system recognizes the discrepancy and asks what you want to do with the mismatching directory by displaying a DIRECTORY NOT CURRENT message on the ODT.

Caution

Serious problems can occur if you mount a disk with an outdated directory and there is not an up-to-date base unit online. The system detects that the up-to-date disk is missing and requests that you mount it. If an OF command is entered as a response, the family is marked online without that disk. The system cannot determine that the directory on the outdated base unit is out-of-date because the system does not have access to the up-to-date disk to compare timestamps.

The system then marks the outdated base unit as even more up-to-date than the offline, up-to-date disk. The file header pointers to areas on the family will be invalid on the outdated flat directory, and changes to the outdated flat directory will make the more current directory also invalid. This problem can never be corrected, and the system cannot detect the conflict.

If the original base unit is deleted from a family on a cataloging system, the original base unit serial number is still used by the volume library to designate the family. This serial number should be specified when a *VOLUME ADD* or *VOLUME DELETE* WFL statement is used for the family.

Duplicating the Catalog File

The AD (Access Duplicate) system command duplicates the system catalog and access structure to protect catalog and volume information. Use of the AD command requires

that the catalog family be a multidisk family. Note also that the family index value specified in the AD command must have a duplicate flat directory. You can also use disk mirroring as an alternative to using the AD command.

A duplicate SYSTEM/CATALOG file is useful on cataloging systems. On noncataloging systems, the SYSTEM/ACCESS file can be rebuilt more easily than it can be duplicated. You can have up to three copies of SYSTEM/CATALOG (the original and two duplicates).

If you use the *SECOPT TAPECHECK=AUTOMATIC* command on a noncataloging system, you can copy the SYSTEM/ACCESS file and make backups from it.

For maximum safety, you might want to have both duplicate online catalogs and backup copies of the catalog. Keeping backup copies of the catalog has one advantage over the use of duplicate catalogs. Changes are made to all online copies of the catalog, and in rare instances corrupted data can be placed in all the copies, such as in the event of an operator error. However, catalog backup tapes have three disadvantages:

- Changes made to the catalog since the catalog was copied to tape are lost when the catalog file is copied back in from tape.
- Lengthy catalog rebuilds and family rebuilds occur when the backup copy replaces the old catalog.
- An interruption in system operation is required to copy the backup catalog back in.

Duplicating the MCP Code File

If a halt/load family has a duplicate flat directory, it also can have a duplicate MCP code file. There are two reasons to have a duplicate MCP code file. The first is to improve system performance; the second is to provide an alternate halt/load unit.

Having two MCP code files on the halt/load family can improve system performance because the files can share the I/O operation load. Performance might not improve significantly, however, if the flat directories of the halt/load family are updated frequently because of the opening, closing, creation, and removal of files. All accesses of files would require additional I/O operations, because more than one directory must be updated.

A duplicate MCP code file also provides an alternate halt/load unit to use if the normal halt/load unit fails. Duplicating the MCP code file to provide an alternate halt/load unit in the current halt/load family is different from making an alternate halt/load family. Consider the following differences between alternate halt/load families and alternate halt/load units when you decide whether your installation will use either or both safety mechanisms, or none at all.

- If errors occur on one copy of a duplicate MCP code file, the system can still use the other copy and does not require a halt/load to recover from the error. A halt/load is required to switch to an alternate halt/load family.
- Alternate halt/load units require duplicate flat directories and a multidisk family, which can slow system performance if the directories are updated frequently or if many files are stored on the family because it has more than one member. Alternate halt/load families do not require duplicate directories and can be single-member families.
- In extremely rare cases, family failures can be so severe that a duplicate MCP code file or flat directory does not help anyway.
- On systems with removable disks, alternate halt/load families can be stored offline; alternate halt/load units cannot be stored offline.

Discussion of this topic is continued under "Making Alternate Halt/Load Families" later in this section.

The CM (Change MCP) system command is used to duplicate the MCP code file. The duplication process results in a halt/load. Up to three disks can contain a copy of the MCP code file (the original and two duplicates), as long as the disks contain a copy of the flat directory.

The name of a duplicate MCP code file is *<file name>/FMLYINX <family index number>*, where the file name is the name of the original MCP code file and the family index number is the 3-digit family index number of the disk that contains the duplicate.

The time needed by the MCP to do the CM operation can be reduced by first copying the MCP code file (giving it the desired name) to the halt/load family members that are to become alternate halt/load units. Unlike the DD and AD commands, the CM command skips the copying operation if it locates matching files with the proper names and timestamps on the proper family members.

Monitoring Duplication

If you have duplicate flat directories, catalog files, or MCP code files, then the duplication operation should be monitored at least once each day or after each halt/load to ensure that all the required duplicates are still active.

It is possible for an operator error or system error to cause the system to stop using a duplicate MCP code file, catalog, or flat directory. The PD (Print Directory) system command can be used to examine the duplication and then restart the operation if one of the duplicate files has become invalid. By issuing the appropriate PD command, you can determine which files are in use. If a flat directory, catalog, or MCP code file is not shown in the PD display as IN USE, then it is not a valid duplicate.

Comparison of Duplication Commands

The duplication of flat directories, catalogs, and MCP code files can be confusing because of misconceptions about the functional or syntactical differences among the files.

The DD, AD, and CM commands each can be used to make up to two duplicate copies of the given file, for a total of three copies. All three commands place the duplicates on different members in the same family. For all three, the file name assigned by the system indicates the family index number of the disk the file is located on, although the form of the name for AD and DD commands differs from that of the CM command. Each copy of the file is always fully contained on one member of the family.

The CM command ensures that the copies match by checking the MCP code file timestamp during the CM operation. Duplicate directories created with the AD or DD commands have special internal timestamps that are checked and changed after every halt/load or every time a disk is readied. Because of an operator error, it is possible for an error in the timestamps to occur after a flat directory is duplicated, so that the timestamps do not match. If that occurs, the system might cancel the AD or DD operation. The system can still use the original copy of the directory, but the AD or DD command must be entered again to restore the duplication process. If the DD operation is canceled for a halt/load family, a duplicate MCP code file for that family is also deleted.

Duplicate directories created by the DD command and duplicate catalogs created by the AD command lead to extra system overhead, because more than one file must be updated every time a file header is added, deleted, or changed. The duplicate MCP code file created by the CM command does not lead to much extra overhead, except that a duplicate MCP code file can be used only on a family that has a duplicate flat directory.

The AD and DD commands require the inclusion of the family index number of the disk that is to receive the copy. The CM command requires a list of the family members that are to receive a copy. If this list is not supplied, the CM command uses the last CM command list supplied for the family. If this is the first time the CM command is used for that family and no list is supplied, the command uses family index 001.

The AD– and DD– commands delete a duplicate copy. The CM– command does not have the same meaning. To delete a duplicate MCP code file, you must enter the CM command with a family index list that omits the family index number of the disk where the duplicate is stored.

The DD and CM commands can be used on any multidisk family. The AD command applies only to the current catalog, so the syntax does not call for a family name or file name.

The AD command can be used on a multidisk family whether or not the DD command has been used on that family. Furthermore, even if the flat directory for that family has been duplicated using the DD command, the AD command is not restricted to placing the duplicate catalog on family members that contain a duplicate flat directory.

The CM command can be used to place a duplicate MCP code file on a family member only if that member contains a duplicate flat directory. However, every member that has a duplicate flat directory need not receive a copy of the MCP code file. For example, there might be two copies of the flat directory but only one copy of the MCP code file.

Making Alternate Halt/Load Families

An alternate halt/load family can be used if the normal halt/load family fails. Making an alternate halt/load family is different from making an alternate halt/load unit on the current halt/load family.

Refer to "Duplicating the MCP Code File" in this section for a description of the differences between alternate halt/load families and alternate halt/load units. You can use either or both safety mechanisms, or none at all.

Using the CM command causes a different copy of the MCP to be executed instead of the one currently running. The system automatically halt/loads when the MCP change is made. If the mix is not null when the CM command is entered (that is, if jobs and tasks are still being executed), then a process called CHANGEMCP appears in the waiting entries and any new processes are scheduled. A scheduled process is one that is temporarily prevented from starting. The change to the new MCP takes place when a null mix is achieved.

If you want the MCP change to take place immediately, then you can use the primitive form of the command (??CM). This command causes a halt/load and changes the MCP immediately, without waiting for a null mix.

Using the RECONFIGURE command regroups the hardware resources of the system. If the NOW option is not included in the RECONFIGURE command, then the reconfiguration does not occur until all groups affected by the reconfiguration have a null mix. While the system is waiting for a null mix, any new processes are scheduled.

The *CM <file name> ON <family name>* command can be used in either of the following ways to make an alternate halt/load family:

- Copy the MCP code file to several families that are usually online and then use *CM <file name> ON <family name>* to designate those families as alternate halt/load families. If the halt/load family experiences a failure, change to one of the alternate halt/load families and run from it while you restore the original halt/load family.
- If you have removable disks, make one or more complete copies of the halt/load family on selected disks. Use *CM <file name> ON <family name>* to designate those disks as alternate halt/load families and then power off those disks and store them offline. If the halt/load family then experiences a failure, retrieve one of the offline families and continue operations with it, including the operation of making a replacement alternate halt/load family.

  This technique can be made even more reliable by rotating these offline backup families as the online halt/load family. This rotation is a way of checking that each backup family is intact. Note that if SYSTEM/ACCESS or SYSTEM/CATALOG file is stored on the halt/load family, time-consuming rebuilds must occur each time the halt/load family is changed.

Note: Do not halt/load using an alternate halt/load family created by either of the methods described previously if the system uses the Mirrored Disk feature. The result is a dead stop–the system halts but does not restart automatically.

The *CM <file name> ON <family name>* command uses the designated MCP code file on the designated family, but it copies a large number of system attributes from the running halt/load family. These attributes cover everything from the host name to the list of saved and reserved units. For this reason, you may want to change a few of these system options (such as CATALOGING or OKTIMEANDDATE) just before entering the CM command. Then, when the alternate halt/load family is finally used as a halt/load unit the system will not be running with the wrong options.

Once an MCP code file has been designated with the CM command, it is marked as a nonremovable file. If you later try to copy in a new version of the file, the copy attempt stops and the DUP FILE(SYSTEM FILE) message is displayed on the ODT. You should enter *CM- ON <family name>* and remove the old MCP code file before starting the copy operation. Once the CM- command has been completed, the disk is no longer a halt/load-capable family until the new version of the MCP code file is copied and the CM command has been used again to designate it as an MCP code file.

If you are using disk drives with removable packs, you might want the backup halt/load families to have the same name as that of the current halt/load family. It is then easier to switch from one halt/load family to another. If you are running on a cataloging system, however, never use the *VOLUME ADD* WFL statement to add a family that does not have a unique family name.

Be sure that you assign a unique family name to any backup halt/load families to avoid conflicts between the name of the backup halt/load family and the active halt/load family. If these precautions are not taken, the system repeatedly issues the error message DUP FAMILYNAME during the copy and CM command operations.

Using the HLDUMPDISK File for Memory Dumps

Memory dumps are taken to either disk files or to tape files. When a memory dump is taken to disk, one of the following two separate dumpdisk files is used:

- The DN file, designated by the DN (Dumpdisk Name) system command. Refer to the *System Commands Reference Manual* for more information on the DN system command.
- SYSTEM/HLDUMPDISK, a special dumpdisk file established by the HLDUMPDISK option of the *CM* system command.

HLDUMPDISK is a special dumpdisk file that resides entirely on the halt/load unit of a system. The name of the HLDUMPDISK file is always in the form SYSTEM/HLDUMPDISK/*nnn* where *nnn* is the family index on which the HLDUMPDISK file resides completely.

An HLDUMPDISK file is established, changed in size, or removed from system use by the CM (Change MCP) system command. System dumps to disk that occur early in initialization before any disk families have been complemented must go to HLDUMPDISK.

Refer to the *System Commands Reference Manual* for more detailed descriptions of the DN and CM commands. Refer to the *System Operations Guide* for more information about dump to disk capabilities.

After dumping memory to a dumpdisk file, the MCP attempts to empty the dump from the file as quickly as possible, so the dumpdisk space is again available for use. You normally empty a dump from a dumpdisk file by converting it to either a MEMORY/DUMP tape or a disk file with the title DP/<date>/<time>/<dump reason>. A DP file is exactly like a raw MEMORY/DUMP tape. When SYSTEM/DUMPANALYZER is run, its file TAPEIN can be equated to the proper DP file.

DUMPDISKMASTER

The DUMPDISKMASTER handles the emptying of dumpdisk files after a dump. DUMPDISKMASTER can be initiated to empty all the dumpdisk files on the system, or to empty a particular dumpdisk file. The specified file might not be currently loaded as the DN file or the HLDUMPDISK file. Only one DUMPDISKMASTER can be working at a particular time. Refer to the *System Commands Reference Manual* for a description of the DN command.

Once running, DUMPDISKMASTER does not terminate until a disposition has been made of each disk dump on each dumpdisk file in use by a system, or until you discontinue it.

DUMPDISKMASTER also handles DN specification changes and the initialization of dumpdisks after a halt/load.

Error Recovery

The disk system can encounter three types of problems:

- Hardware problems, such as a broken disk drive or a damaged or destroyed disk
- Operator error, such as the accidental reconfiguring of a disk or the accidental removal of files
- Directory software errors, such as directory I/O errors, directory data corruption, or MCP program errors

Although these problems are rare, you must be able to recognize them so that they can be resolved quickly with little or no loss of data. You, or your operations staff, should know the following:

- How to isolate defective sectors on a disk
- What to do if base units or continuation units are damaged or destroyed
- When disks can be moved from one drive to another to eliminate errors
- When data can be moved from one disk to another to eliminate errors
- What to do if directory errors are encountered

- What to do if errors are encountered during family rebuilds

Once the disk label and flat directory header have been read successfully, the system provides various techniques for overcoming errors.

Isolating Defective Sectors

When an I/O error occurs while the system is using or trying to use a disk file, the entire disk is not necessarily damaged. A group of sectors on the disk might be defective, and it is possible to isolate these bad sectors so that the rest of the disk can still be used.

When the system encounters an I/O error, it automatically attempts the I/O operation again and places an entry in SYSTEM/SUMLOG that describes whether the error was corrected in the retry or whether the error was not corrected and the I/O operation was discontinued. These log entries and the I/O error messages that are displayed on the ODT can be used to determine if the problem is caused by a few defective sectors or if the entire disk is damaged.

If the log entry or the I/O error messages specify a certain range of sectors repeatedly, the damage is probably limited to those sectors. Also, if the I/O error affects only a few files, but the rest of the files on the disk are accessed properly, that is an indication that the problem was caused by a few defective sectors.

You can use the SCAN (Scan Disk or Pack Volume) system command to read a disk and record any defective sectors that were encountered during an attempt to read a file on the disk. The SCAN operation is a time-consuming process, and should be done at night or another time when system usage is minimal.

The RES (Reserve) system command transfers data stored on defective sectors to other sectors. The AS BADDISK clause of the RES command causes the system to mark defective sectors so that they are not used again.

Damaged or Destroyed Disks

A disk is considered damaged if it experiences irrecoverable I/O errors or if the directory is corrupted. A disk is considered destroyed if it is physically broken. There are different ways to resolve the problem of a damaged or destroyed disk, depending on whether the disk is a base unit or a continuation unit.

Replacing a Base Unit

If the base unit of a disk family is destroyed and the flat directory is not duplicated on another family member, you must use the RC (Reconfigure Disk) system command to reconfigure a new family and then copy all the required files back onto the family from the latest backup tapes. You cannot reconfigure a new base unit in place of the old one, even if the same serial number is used, because the flat directory on the old base unit is lost.

The system marks all continuation units in a family with the date and time that the base unit was reconfigured. When you reconfigure a new base unit with the RC command, there is no link between the new base unit and the old continuation units. If the system did not take this precaution, continuation units from an old family would get confused with continuation units for the new family. You can check the creation date and time of a disk family with the OL (Display Labels and Paths) system command.

If you are running a cataloging system and need to reconfigure a new base unit, you must take special steps to avoid losing all the catalog information for the old family. The system asks if the new family should inherit all the backup information for the old family. This request is made by displaying the following message on the ODT:

```
PK <unit number> OK TO RE-ENTER INTO VOLUME LIBRARY
```

A reply of OK retains the catalog information. Refer to "Replacing a Damaged Volumed Disk" in Section 6 for more information.

Replacing a Continuation Unit

If a continuation unit is destroyed, there are more ways to replace it than there are for a base unit. The base unit with the flat directory is still intact, so files on the base unit and other continuation units are still usable. When programs try to access files that are partially or totally stored on the missing continuation unit, the system displays the following RSVP message:

```
REQ·<family name> <serial number>
```

An RSVP message is one that requires a response by the system operator. In this case, the operator must discontinue the programs by using the DS (Discontinue) system command.

To create a substitute continuation unit, use the RC (Reconfigure Disk) system command. When you use this command, the system replaces the old continuation unit with the new one and removes all the files in the entire family that had portions stored on the old continuation unit.

To retain files that had portions stored on the missing continuation unit, add the KEEP option to the RC command. In this case the system does not completely remove the files, but marks them as having some areas missing. Programs then can use some of the data in the files, but not data from the areas that were on the missing continuation unit. If a program tries to access data from the areas that were on the missing continuation unit, an I/O error occurs.

Moving Disks to Another Disk Drive

When a disk drive breaks down or appears to be receiving many I/O errors, it might be possible to correct the problem by moving the disk pack to another drive. This technique applies only to removable media such as model 677 disks; you cannot move nonremovable media such as model 207 disks.

Caution

Before moving a disk from one drive to another, especially a disk that has experienced errors, have a Unisys field engineer inspect the disk to make sure that the surfaces of the disk have not been marred or scratched. A scratched disk can, in turn, scratch the disk drive read/write heads. Scratched read/write heads can then scratch other disks mounted on them, and so on.

It is useful to keep a written log of the drives that disks have been mounted on, by time and date. Then, when a scratched disk surface or a scratched disk read/write head is discovered, you will know which other disks and drives to inspect for damage.

Locate an available disk drive. This unit must be acquired and not saved, not reserved, and not in use by another disk. If the disk that you want to move is not in use, all you have to do is power off the disk and move it to the other drive. If, however, the disk that you want to move is in use, then you must use the MOVE (Move Job/Pack) system command. The system does some checking and then displays a message requesting you to move the disk manually.

On systems other than the A 12, A 15, A 16, and A 17, some families (such as the family on which the JOBDESC file is stored) cannot be moved by this technique. On A 12, A 15, A 16, and A 17 systems, any disk can be moved by using the MOVE command.

Sometimes the I/O error problem occurs so many times that a halt/load is necessary before you can move the disk to an operable disk drive. However, you might be able to avoid the halt/load by performing the following steps. (This process is not recommended for A 12, A 15, A 16, and A 17 systems.)

1. Halt the processors.
2. Manually move the disk.
3. Start the processors when the disk is ready.

This technique does not always work. If the system does not start running again, proceed with a halt/load.

Moving Data to Another Disk

Usually when a disk experiences problems, it is not totally unusable. If a particular disk in a family produces a large number of I/O problems, you can move the data to another disk by one of two techniques:

- If the disk is a continuation unit in a multidisk family, use the RES (Reserve) system command to reserve the disk.

  The system copies all the data from the disk to other members in the family. Once the reserve operation is completed, you can perform maintenance operations on the disk; the PG (Purge) system command can be used, for instance. If parity errors are encountered during the reserve operation, the operation terminates and displays a message on the ODT that specifies which addresses encountered the error. You must then decide whether to remove the affected files or use the COPYERRORS clause of the RES command. COPYERRORS causes the reserve operation to copy all the data on the disk, including data that has parity errors. The data that has parity errors is still invalid and cannot be used.

  This version of the RES command also can be used to take a disk out of a family even if the disk is not experiencing I/O problems.

- Use the REPLACE (Replace Disk or Pack Volume) system command instead of the RES command in the following situations:

  - If the disk is a base unit or if it contains a large amount of data. In this case, the reserve operation can be a lengthy process. Also, if the disk contains a large amount of data, it can be difficult for the system to find sufficient space for the files on the other members of the family.
  - If you know that the disk contains portions of files that were assigned to the disk with the FAMILYINDEX file attribute. The reserve operation does not permit files to be moved from a disk if the files were assigned to that family member with the FAMILYINDEX attribute.

  After issuing a REPLACE command, the system then copies all the files and label information from the original disk to the new disk. The new disk receives the serial number and family name of the old disk. When the copying is complete, the system erases the label on the old disk so that it does not become confused with the new disk.

  Note: The REPLACE system command cannot be used to make backup copies of disks because it erases the label of the source disk.

Directory Error Recovery

Disk directories are very important. If something goes wrong with a directory, then it is possible to lose some or all files referenced by that directory. The directories are designed so that errors in one part of the directory do not adversely affect other parts of the directory. The system constantly monitors the directories for discrepancies. When an error is detected, the system automatically attempts to eliminate the problem. The following text describes the system features that help to ensure the integrity of the directories.

Standard Disk I/O Error Recovery

The standard I/O subsystem attempts to recover from all failed I/O operations, including directory I/O errors. Only I/O errors that are not corrected by these attempts will cause problems.

Directory Record Integrity Tests

The MCP tests all directory records before they are used. Each record must pass tests such as HDRMARKER, HDRLOCATION, and CHECKSUM. If a record fails one of these tests, the directory subsystem invokes one or more of the following procedures: error reporting, recovery, or termination.

Automatic ERRORHANDLER Family Rebuilds

These rebuilding procedures are invoked to build a new File Access Structure Table (FAST). Rebuilding the family solves two types of problems. First, certain types of bad records in the flat directory or catalog are bypassed when the family is rebuilt so that these records are not referenced by the rebuilt FAST and do not cause problems. Second, certain corrupted FAST records are discarded during the rebuilding process. Refer to "Errors Occurring When Families Are Rebuilt" later in this section for more information.

Directory Duplication

Duplicate directories on multidisk families can be used to solve most directory I/O errors and problems caused by directory data corruption. Duplicate directories also prevent directory loss if a disk is physically destroyed. Duplicate directories require a modest amount of additional disk space and require more I/O operations because more than one copy of the directory must be updated.

Types of Directory Error

Disk directories are stored as disk files. As with any disk file, they can experience problems. The flat directory and the catalog can have three different, but interrelated, types of problems:

- Directory I/O errors
- Directory data corruption
- MCP program errors

These types of errors are very infrequent. Their characteristics are described in the following paragraphs.

Directory I/O Errors

If a disk I/O error message immediately precedes or follows a directory error, then you can infer that the I/O error is the root of the directory error. You can also make finer distinctions. If the I/O error occurred during a write operation, then it indicates that a file header that was just added, removed, or changed in the directory might not be usable the next time it is referred to. If the error occurred during a read operation, then it indicates that the system was attempting to access a directory record (such as a disk file header) and that directory record is unreadable. Repeated references to that same file presumably would cause repeated read errors.

Some I/O errors can be easily corrected. For example, you might be able to solve a Not Ready or Write Lockout error by turning the proper switch to the proper setting. On the other hand, parity errors on read operations usually are not correctable. In fact, the system only reports a parity error if several attempts to perform the read operation fail. Correction of parity errors depends on the exact nature of the error. If the data truly has a parity error, then that directory record is lost. But if the parity error is caused by a faulty disk drive or a disk drive controller malfunction, fixing the hardware restores access to the directory record.

Directory Data Corruption or MCP Errors

It is difficult to distinguish directory data corruption from MCP errors. A typical directory data corruption error might cause a CHECKSUM, HDRMARKER, or HDRLOCATION error message. Note that all read errors also appear to the MCP directory subroutines as directory data corruption, so read errors also cause error messages for CHECKSUM, HDRMARKER, HDRLOCATION, and so on. On the other hand, a typical MCP error results in a memory dump by FILEHANDLER. These are only tendencies, however. MCP errors can cause HDRLOCATION errors, and data corruption can cause FILEHANDLER memory dumps.

When the system detects a directory error, it reports the problem with one or more messages. Often, the system will then attempt some type of automatic error recovery. The system usually reports the progress or failure of the recovery with additional messages. These various messages are clues as to what happened. You should try to answer the following list of questions after a directory error occurs. The answers to these questions can help you decide what further corrective actions you might need to take.

- Did the system read the disk label and open the base unit directory, or did the system issue an error message indicating that the disk cannot be used? If such an error message appears, there is a fundamental problem with the disk, and not a problem with individual files on the disk.
- Does the problem appear to affect only one or a few programs or files? Does it appear to affect every file on a family? Does it appear to affect every file on the system? These questions can help you isolate the problem. If only one or a few programs or files encounter errors, the problem usually is within these programs or files. If only one family encounters errors, the problem is within that family.
- Does the problem appear to be in the flat directory on a given family, or does it appear to be in the catalog or access structure? Such problems are usually caused by I/O errors. If the word FLAT appears in the error message, then it indicates that the flat directory might have a problem. If the words PAST, FAST, CATALOG, VAST, or VOLLIB (Volume Library) appear, it almost always means there is a problem in the catalog or the access structure.
- Does the problem appear to be an I/O error, or does it appear to be a case of data corruption or an MCP error? If it appears to be an I/O error, try to correct the problem as discussed above. If it appears to be a case of data corruption or an MCP error, contact your Unisys field engineer or file a User Communication Form (UCF) with Unisys.

- Does the problem appear to be intermittent, or has the problem been there for a long time? Does the same kind of problem occur occasionally for different files, or does the problem appear only for a particular test case? If you can isolate the test case, you can show it to your Unisys field engineer.
- Does the problem go away after rebuilding the family? If it does, then the rebuild probably took care of the problem.

Be sure to locate the original error message, in SYSTEM/SUMLOG if necessary. Often one error triggers several error reports. The first error message can be the best clue. The LOGANALYZER utility is often helpful in solving I/O error problems because all I/O errors and an audit of any retry attempts are recorded in SYSTEM/SUMLOG. You can use LOGANALYZER to examine the errors of a particular disk by entering *LOG MAINT PK <unit number>*, where <unit number> is the unit number of the disk drive on which the disk is mounted. LOGANALYZER then prints out a list of all I/O error retries and messages that have occurred for that disk drive since the log was initiated. These I/O error messages can help you identify the problem, and you can then contact your Unisys field engineer if necessary.

Errors Occurring When Families Are Rebuilt

A family can be rebuilt for three reasons:

- If necessary, when a base unit is first placed online
- When the RB (Rebuild Access) system command is invoked
- For error recovery

On cataloging systems, rebuilding of the catalog file also occurs during error recovery. Rebuilding a family and the catalog normally do not occur on a cataloging system when a volumed disk is placed online.

Rebuilding a Family for a New Base Unit or Halt/Load Unit

A family is rebuilt when a base unit is first placed online or when a halt/load occurs. Regardless of whether the family needs to be rebuilt, the system reads the entire flat directory on the disk to determine what sectors of the family are already allocated to files and what sectors of the family are available for new files. If the family needs to be rebuilt, then the FAST is rebuilt at the same time.

The MCP issues error messages if any errors are detected during this reading of the flat directory or when the FAST is rebuilt. Immediately after the first such error message, the MCP displays the following message on the ODT:

```
OK TO ERASE BAD RECORDS?
```

This message indicates that the MCP has just detected an error; there might be more errors as the rebuilding operation progresses. The MCP is asking for permission to erase all flat directory records that caused errors. You can enter one of three replies: DS, OF, or OK.

If you enter the reply DS, processing of the flat directory halts and the system marks the disk offline.

The OF reply to the OK TO ERASE BAD RECORDS? message is the safest reply. An OF reply causes the rest of the directory to be processed, but does not cause the bad records to be erased.

The advantage of the OF command is that it gives the system a chance to process the rest of the directory without accidentally erasing it. For instance, if the cause of the original problem was that the disk was being read on a faulty disk drive, then erasing directory records could erase good records from the directory.

When you enter an OF command, the system marks the disk as not having any available space for new files. This feature prevents a new file from accidentally being allocated in areas of the disk that overlap the areas in use by the files that apparently were bad. If you need to add records or files to the disk, you must remove existing files to make room for the new records or files.

After the system reports one bad record in the flat directory, the system often reports a few subsequent consecutive directory records as also being bad. Directory records are variable in length and several can be accessed together in one block, so additional bad records are often all part of the original bad record. If only a few extra bad records are reported, you usually can assume that there is actually just one bad record.

You can enter an OK command in reply, which erases corrupted directory records. This reply could lead to the loss of some files on the disk. Use this reply unless you suspect that the disk I/O hardware is defective and might be causing the problem. If you enter an OK reply, the system proceeds with rebuilding the family and then marks the disk online.

If the rebuilding operation is being done while readying a disk or halt/loading, you can terminate the operation by entering the CL (Clear) system command with the syntax *CL PK <unit number>*. The system stops the rebuilding operation and marks the disk offline. This can be a useful option if the disk becomes NOT READY during the rebuilding operation or you decide the wrong disk was readied.

Using the RB (Rebuild Access) System Command

If you need to build a new access structure, you can initiate the operation by entering the RB (Rebuild Access) system command. You can use the RB command when you think that the access structure is corrupt or does not actually describe the files on a disk.

The rebuilding operation initiated by the RB command only reconstructs the FAST; it does not build an available disk table, does not ask the operator questions, and does not erase flat directory records (good or bad). Also, this rebuilding operation cannot be cleared by the CL (Clear) system command.

Rebuilding a Family during Directory Error Recovery

A family is rebuilt when the directory error recovery procedure ERRORHANDLER is invoked. ERRORHANDLER is invoked automatically when an error is detected while accessing the directory of an online family.

This rebuilding operation only reconstructs the FAST; it does not build an available disk table, does not ask the operator questions, and does not erase flat directory records (good or bad). Also, this type of rebuilding operation cannot be cleared by the CL (Clear) system command.

Mirrored Disk

On A Series systems, the Mirrored Disk feature enables you to maintain from two to four disks as a mirrored set, that is, as exact copies of each other.

The Mirrored Disk feature increases both system availability and data integrity. If one copy in a mirrored set is destroyed, goes offline, or experiences any irrecoverable errors, another online copy in the set will allow normal functioning to proceed. Disk mirroring is completely transparent to applications. If an error occurs on one mirrored disk, the application proceeds normally, and the system operator is notified of the error.

Mirroring is also useful in enabling you to relocate disks to improve system performance without disrupting normal system operations. To achieve this, you create a mirrored disk using the MIRROR CREATE command, then release the original disk using the MIRROR RELEASE command. Refer to "Operational Information" later in this section.

For mirrored disks, you should reevaluate the site's backup and audit features or procedures. Because mirroring significantly decreases the possibility of data loss due to equipment or media malfunction, the number of available processing hours on a system is increased.

Mirroring disks can improve I/O throughput on disk subsystems that experience a high ratio of reads versus writes. This improvement occurs because the MCP distributes the read operations equally to all the mirrored disks in a set. If a disk subsystem has a low ratio of reads to writes, the I/O throughput might be reduced, because writes must be issued to all mirrored disks in a set. The actual impact of mirrored disks varies according to the individual characteristics of the installation.

Mirrored copies of existing disks can be created without bringing disks offline or interrupting use by the system.

No special hardware is required for disk mirroring. However, you must have available disk units to allow for the redundancy. The Mirrored Disk feature applies to all supported disk types and can be used with both cataloging and noncataloging file systems.

Mirrored Disk Options

The Mirrored Disk feature enables you to specify which disks are mirrored, the number of disks in a set, and the units on which mirrored disks are to reside. Disks within a given mirrored set must be of the same type. Refer to "Configuration Recommendations" later in this section for system configuration guidelines and for the location of mirrored disks within a set.

Once created, mirrored sets are maintained automatically across system interruption. Operator action is required only in certain exception conditions.

You can remove mirrored disks from any mirrored set or create new mirrored disks for any set while the system is in normal operation and while the disks are in use. You can also move mirrored sets between systems and within systems.

Recovery options for any mirrored set are specified with the MIRROR OPTION system command. This command determines the action to be taken after a halt/load if certain critical MCP information has been lost.

Mirrored Disk Initiation

The Mirrored Disk feature is initiated by setting the MIRRORING system option, then halt/loading the system; this creates the internal structures needed for mirroring. Thereafter, mirrored copies of existing disks are created using the MIRROR CREATE system command, and all other MIRROR commands will be valid.

The MIRRORING system option must be set when an existing disk subsystem is moved to a new system, or when a system is cold-started with preexisting mirrored sets. Once the MIRRORING option is activated, mirrored disks are brought online following the next halt/load.

Creating Mirrored Disks

Once mirroring is initiated, mirrored disks are created with the MIRROR CREATE system command. This command copies one or more source disks and brings the destination disk online. The source disk can be a part of an existing mirrored set. The source disk remains online during the copy process.

The destination disk must not have any open files on it and must not be a member of a mirrored set when the MIRROR CREATE command is issued. All files on the destination disk are overwritten during the mirror creation process. Therefore, you should copy any files that you want to save from this disk to another disk before creating the mirrored disk.

Configuration Recommendations

To ensure full redundancy of disks, Unisys suggests that separate paths be maintained to disks within each mirrored set. If the system is configured in this way, the MCP can handle disk failures as well as failures of controllers and other hardware. Maintaining

separate paths to mirrored disks within a set is not required by the Mirrored Disk feature; the decision is up to you. Other configuration options, such as multiple paths to disks through different controllers and exchanges, can be just as beneficial. Refer to the *System Configuration Guide* for full details about configuring an A Series system.

Note that the mirroring feature creates a redundant disk device. The entire disk system can be made redundant by the addition of more data link processors (DLPs), controllers and exchanges.

Mirrored Disk I/O Handling

I/O operations are handled differently for mirrored disks and nonmirrored disks. Read operations are issued to only one disk within a mirrored set, while write operations are issued to all disks within a mirrored set. (ALGOL SEEK operations are considered to be read operations.)

In the following discussion, *online* refers to current members of a mirrored set that are available and synchronized; *offline* refers to disks that are in a state of Write Lockout, are not ready, or are otherwise unavailable to the system because they are being audited. The term *pending disks* refers to disks returning from an offline to an online state, mirrored set members being created, and all members of a mirrored set containing members not yet visible to the system. All three states are mutually exclusive.

Read Operations

Reads are issued to each online disk in turn within a mirrored set. If an I/O error occurs on a read operation to a disk, the next disk in the mirrored set is selected and the read is issued to that disk.

Not Ready and Write Lockout errors cause the disk to be placed in an offline state, and an audit is started for the disk. Any other read errors release the disk from the mirrored set. If errors occur to all online disks within a set, the application program receives an error on the I/O operation; otherwise, the errors are transparent to the application. Offline and pending members are not candidates for reads.

Write Operations

Write operations are issued to all online disks within a set. If an irrecoverable write parity error occurs, the disk is released from the set. For Not Ready and Write Lockout errors, the disk is marked offline, and an audit for the disk is started. In either case, the operator is notified by a message at the ODT and appropriate action should be taken. An application program can receive an error on a write only if errors are encountered by all online disks within a set.

If an irrecoverable write error occurs on a mirrored disk, the disk is released from the mirrored set, and a message is issued indicating that the release has occurred. The disk label is modified such that the disk cannot be used without reconfiguring the disk with the RC system command. In this situation, you should resolve the problem and decide whether a new mirrored disk should be created for the set.

If a disk goes offline, an audit is automatically started for the disk. A message is issued identifying the disk as being in Write Lockout or Not Ready.

Whenever a write is directed to a mirrored set, it is recorded in an MCP table called the Outstanding Write List (OWL). When the writes to associated disks are complete, the entry is removed from the OWL. After a halt/load, the OWL updates disks within a set so that they are identical.

If the OWL is destroyed, the disks might not be identical, because the writes in process to a set might not be complete for all members of the set. The MIRROR OPTION system command allows you to specify appropriate action in the event the OWL is lost.

Audits

An audit table of write operations is kept for offline mirrored disks. This audit table is a bit map of the disk in question. Each bit in the table corresponds to a fixed area on the disk. Therefore, the size of the audit table is fixed and is a function of the number of cylinders defined for the disk type. Whenever a write is issued to a mirrored set, corresponding bits are set in the audit table of each offline member.

The audit table is abandoned if the offline disk is released from the mirrored set. Disks being audited at the time of a halt/load are released from the mirrored set.

When an offline disk is returned to an online state, the audit table is used to update that disk.

Operational Information

Mirrored disks can be moved within a system just like nonmirrored disks. Mirrored sets also can moved between systems and MCPs.

Moving Disk Units within a System

You can move members of a mirrored set within a system while it is down with no ill effects.

For removable media, either use the MOVE system command or power off and move the disk to a new drive. In either case, the disk being moved is audited just as any other offline copy. The rest of the mirrored set remains online and usable. After you have moved the disk to the new drive, the audit is applied to update the disk.

For fixed media, you can copy data to other units by using the MIRROR CREATE system command. After the new copy is complete, you can release the source disk by using the MIRROR RELEASE system command.

Moving Packs between Systems

It is possible to move mirrored sets between systems by physically removing them from one system and installing them in another. Before you move the set, however,

you must use the CLOSE and FREE system commands for the disks in the set. The system updates the disk labels to indicate that the disks are closed. After being moved, the mirrored set must be readied using the RY system command. When the disks are readied by another system, the new system reads the label to determine if the disk was closed before moving and to determine the number of valid copies in the mirrored set.

If sets that were not closed are moved to a new system, they are not brought online as a set unless their recovery type was specified as *DMS* with the MIRROR RECOVERY command. For more information, refer to the *System Commands Reference Manual.*

Mirrored sets with a recovery type of DISCARD are broken (mirroring is discontinued) when one disk is brought online as a nonmirrored unit. The other members of the set are released (the disk label is invalidated, the disk is not brought online, and a message is issued).

Mirrored halt/load units cannot be used to halt/load on another system. This action results in a *deadstop,* meaning that the system halts but does not automatically restart.

Bringing Offline Disk Packs Back Online

When you bring a mirrored disk online, the system verifies that mirroring has been established for that system. If mirroring has not been established, the following message is displayed and the process of bringing the disk online is terminated:

```
PK<unit number> [<serial number>](<family name>) ERR: MIRRORED PACK
  ON NONMIRRORED SYSTEM.
```

You can bring a mirrored disk online as a nonmirrored disk only by issuing an RC (Reconfigure Disk) or PU (Purge) system command for the closed unit.

When a mirrored disk is readied on a mirroring system, one of three conditions is required:

- The disk must have been closed prior to going offline.
- The system must have a record of the disk in question.
- The disk must have a recovery type of DMS.

If the disk was closed, the disk is brought online when the other members are visible to the system, or when audits are begun (with the MIRROR AUDIT system command) for those members not yet visible to the system.

If the disk was not closed, but the disk is recorded in the system tables, the system does one of the following:

- If a halt/load occurred while the disk was offline or the disk encountered a parity error, the disk is released (that is, the disk label is invalidated and the disk is not brought online) and a message is issued.
- If the disk went offline and is currently being audited, the audit is applied to the disk. When the audit is complete, the disk is returned online.

- If the disk was online and is returning after a halt/load, it is brought online when the other members are visible to the system. Any writes that were outstanding at the time of the halt/load are applied.
- If the OWL was corrupted or lost during the halt/load, the set is brought online only if it has a recovery type of DMS; otherwise, only one disk of the set is brought online as a nonmirrored unit, and the other members are released.
- If a mirrored disk returning online has no history of being present on the current system (or the disk has been in use on another system) and the set was not closed (by means of the CLOSE system command), then the set is brought online only if its recovery type is DMS.
- Similarly, if the OWL is lost during a halt/load, only those sets with recovery type DMS are brought back online. When these situations arise for sets with recovery type DISCARD, only one disk is brought online as nonmirrored, and the other members are released.

Following a halt/load (or when a mirrored set is moved to a new system), a set does not go online until all disks previously online (or previously part of the closed set) are visible to the system. If some of the members are absent, or you do not want to wait for all the members to be ready before using the set, the absent members can be forced into an audit state and the set brought online by means of the MIRROR AUDIT command.

Sets that are in the process of being brought online are called partial mirrored sets. If a task is initiated that causes a list of partial mirrored sets to be displayed, and if that task becomes a waiting entry, the list of partial mirrored sets is redisplayed whenever a change occurs in the state of a mirrored set or when the system operator reactivates the waiting entry using the OK system command. Partial mirrored disks are identified in the PER and OL system command displays with a lowercase *m*.

Recovery

When a system is halt/loaded with mirrored critical units (that is, units required for the system to run), offline copies of those units are automatically audited until they are visible to the system. This allows the system to start and you to take appropriate action if one of the members is missing or corrupted before, during, or after the halt/load. Noncritical units are handled as described earlier under "Bringing Offline Disk Packs Back Online."

Note: *Mirrored halt/load units cannot be moved between mirrored systems. The result is a dead stop.*

During a halt/load, all needed structures are created or recovered. The Mirror Information Table (MIT), which contains information concerning all mirrored sets on the system, is stored on the halt/load disk as an MCP structure and is restored from there after a halt/load. The OWL, which keeps track of writes in process to mirrored sets, is preserved in memory. If this structure is corrupted or lost during a halt/load, only mirrored sets with the DMS recovery type are brought online after the halt/load. Other mirrored sets are broken and only one member is brought online as a nonmirrored member.

The audit tables for offline disks are not preserved across halt/loads and the offline disks are released from the set when they are brought online. Disks having their audits applied and incomplete online creations are also released upon return to service.

Deallocating Mirrored Disks

You can deallocate mirrored copies from a mirrored set using the MIRROR RELEASE system command. Different command forms allow you to release a specified disk, to release offline copies of a specified disk, and to release all copies of a specified disk. During normal system operations, you can release copies at any time.

Mirrored disks being audited, created, or having audits applied are automatically released from sets after a halt/load. Units receiving parity errors are also released if they are not the last online member of a mirrored set.

Whenever a disk is released from a mirrored set as a result of an operator command, a system error, or loss of the OWL, the disk label is invalidated, thus requiring an LB (Relabel Pack or Host Control Unit), a PG (Purge), or an RC (Reconfigure Disk) system command before the disk can be used again. If the label cannot be updated at the time, the label is invalidated when next visible to the system. This feature is necessary to prevent out-of-date members from replacing more recent copies of a unit.

Caution

If you ready out-of-date members on another mirroring system, these members can be brought online. This action could corrupt data, so take special care to prevent this from occurring.

Transferring MCPs

When you create an alternate halt/load family using a *CM <file name> ON FAMILY* system command, system mirror information is copied to the disk for which the MCP has been changed. Any mirrored set manipulation between the time the MCP change is performed and the time of the subsequent halt/load will invalidate the linkage between the mirror information kept in memory and that of the newly changed unit.

Invalidating manipulation includes the following:

- Using MIRROR system commands
- Bringing mirrored units online
- Releasing mirrored units owing to exception conditions
- Auditing mirrored units owing to exception conditions
- Closing a mirrored set

If such a situation arises, the result is a dead stop when the halt/load is performed. To resolve this situation, you must halt/load back to the original halt/load unit. In general, avoid changing halt/load units when mirroring unless absolutely necessary.

A similar situation arises if you attempt to transfer halt/load units between mirroring systems. This is not allowed and always results in a dead stop.

Precautions

The MIT on the halt/load unit keeps track of the status of the mirrored disks on a system. It also contains records of disks that have been removed from a mirrored set, but whose labels could not be invalidated. These records are the only protection against such an outdated disk returning online. As a result, if the MIT is lost (for example, by changing halt/load units), that protection is removed. Data corruption can occur if you return such an outdated disk online. Therefore, keep the following recommendations in mind:

- Instead of maintaining a separate backup halt/load unit, make a mirrored copy of the current halt/load unit. This ensures that the MIT is not lost.
- If you choose to maintain a separate backup halt/load unit, update that unit using the CM (Change MCP) system command every time a disk is added or removed from the set of mirrored disks currently online. Update the backup halt/load unit in the following cases:
  - When a disk unit is added to a mirrored set
  - When a disk unit is released from a mirrored set
  - When a preexisting mirrored set is brought online
  - When a mirrored set is closed
  - After a halt/load
- If a dead stop occurs, halt/load back to the original halt/load unit. If this cannot be done (as when the original halt/load unit has failed), warm-start the system using the LOADER program. In this case, the MIT is lost. Be very careful not to bring any outdated disks online.

Memory Disk

The Memory Disk enables you to use memory as if it were a disk unit. Memory Disk provides file access with extremely high data-transfer rates and relatively little access time. You can configure a system with one or two units of Memory Disk, each with up to 10 megawords of memory (60 megabytes).

The units can be used, with certain restrictions, in the same way as any other disk. A Memory Disk family can be used like any other family with the following exceptions: it cannot be the halt/load family, and it cannot contain a dumpdisk file.

The unit designations are retained across halt/loads, and files on the Memory Disk units are retained across most halt/loads. Vulnerability of data is discussed later in this section.

Memory Disk is supported on all A Series systems.

Memory Disk is established when you use the RECONFIGURE (Reconfigure System) system command to reconfigure to a group that contains a Memory Disk declaration. Instructions on how to make a Memory Disk declaration are provided in the *System Configuration Guide.*

The system allocates the requested amount of memory (or as much memory as is available) and fills Memory Disk with zeros. The location of the allocated memory is preserved on the halt/load unit so that the Memory Disk areas can be found after a halt/load. Each Memory Disk unit is reconfigured and brought online automatically if the family name is supplied in the declaration. Unnamed Memory Disk units must be reconfigured manually with the RC (Reconfigure Disk) system command.

Vulnerability of Data

Memory Disk offers high-speed file access in a convenient and flexible manner, but memory is a volatile medium that is vulnerable to power failure and memory reconfiguration. Therefore, you should carefully choose the files that are to reside on Memory Disk. You will have to reconstruct those files after any system event that causes memory to be altered or corrupted.

Possible candidates for placement on a Memory Disk unit are read-only data files, code files, and temporary files. You might also consider using Memory Disk for the DL SORT system command and your data comm information files. If you are not using the cataloging and InfoGuard tape security subsystems, you might also consider using Memory Disk for the SYSTEM/CATALOG file. If you are using the cataloging and InfoGuard tape security subsystem, the risk of losing the catalog probably outweighs the performance benefits of Memory Disk.

Creating a Memory Disk Unit

You create a Memory Disk unit by modifying the configuration file for your system. You add a new declaration in the MEMORY section of your system group. The declaration specifies a unit number and the amount of memory associated with the unit. You can optionally specify the family name or base unit of the family.

The amount of memory can be specified as a number of words. The amount actually used by the system is the nearest lower multiple of 512 kilowords.

For specific information on modifying the configuration file, refer to the *System Configuration Guide.*

Initializing Memory Disk

Memory Disk is allocated above the system memory limit during memory establishment. A warning message is issued if a unit has insufficient usable memory. The amount and location of the memory are saved to allow the unit to be reestablished after a halt/load.

The Memory Disk units are established during peripheral initialization. Each unit is assigned the identifier *DK <unit number>* and is defined as a disk device with a maximum of 10 modules and with 2 switches per module. Each module contains the equivalent of 34,952 disk segments (approximately 1 megaword). Each switch contains the equivalent of 17,476 disk segments (approximately 512 kilowords). The size of a Memory Disk is always an integer number of switches.

If you specified a family name for the Memory Disk unit in the configuration file, the system automatically reconfigures and readies the unit whenever necessary. This process happens on the next halt/load after you reconfigure the system or change the memory configuration of the system. If you did not specify a family name for the Memory Disk unit in the configuration file, you must manually reconfigure the Memory Disk unit whenever necessary by using the RC (Reconfigure Disk) system command.

Memory Disk Halt/Load Recovery

All memory used by Memory Disk is reestablished from information saved on the halt/load disk. The memory areas marked for Memory Disk are not overwritten by the processors during memory test. If the current memory configuration does not match the saved configuration or if the Memory Disk areas have been overwritten, all the Memory Disk areas are marked as available, the saved information is removed, and an error message is displayed. The Memory Disk configuration entry is then in its initial state. Memory is reallocated as described earlier under "Initializing Memory Disk," and the unit is reinitialized.

Memory Reconfiguration

Some memory reconfiguration commands cause Memory Disk to be reinitialized at the next halt/load. The system issues an appropriate request for response (RSVP) before the reconfiguration is performed. If you elect to proceed with the memory reconfiguration, you should be aware that the next scheduled or unscheduled halt/load will cause Memory Disk to be reinitialized.

I/O Handling

Memory Disk I/O operations are billed in the same way as other I/O operations except that the processor time used by the data transfer procedure is charged to the task as processor time only. The bytes transferred are included in the MCP or user utilization counts as appropriate. The bytes are also counted into separate Memory Disk MCP and user subtotals. Because Memory Disk operations are synchronous (no I/O interrupt occurs), they do not contribute to either the IOFINISH or IOINTERRUPTS system counts.

Any irrecoverable error that occurs during a Memory Disk operation is reported in the logical result descriptor. The normal system action occurs for correctable memory parity errors, so they are recoverable. All other errors are irrecoverable and are not retried. Memory parity errors and memory fail errors are reported as I/O parity errors but are also logged normally as memory errors. Missing pages are reported as NOT READY I/O results. The memory protect bit of the Input/Output Control Word (IOCW) is honored

only if the MCP is compiled with the DIAGNOSTICS option. Memory protect errors are reported as descriptor errors.

Operational Restrictions

The following restrictions apply to a Memory Disk unit:

- It cannot be the halt/load unit.
- It cannot contain a dumpdisk file.
- It cannot be specified in a FREE (Free Resource) system command.
- It cannot be a member of a family that is not a Memory Disk family.
- Peripheral test driver (PTD) tests cannot be run against it.

Operational Consideration

When using Memory Disk, this consideration should be taken into account: a system dump does not dump the contents of Memory Disk.

Section 6
Protecting Disk Files

Protecting information stored in disk files is one of the primary concerns in managing any computer system. Means of doing this include making extra copies of files and storing them on offline disks, or making copies of files on tape and then keeping the tapes onsite or in a secure location away from the site to prevent loss from fire or other accidents.

Keeping track of the backup copies of different versions of a file can be a complex task. Unisys provides two ways to track and coordinate the creation and maintenance of backup file copies on A Series systems: the cataloging system, and the archiving subsystem.

Both of these subsystems use the MCP library maintenance procedure to make backup copies of disk files and to restore those copies to disk. Both of these subsystems use special directories to keep track of the backup copies.

The cataloging system supports tracking backup copies of several generations of files. Tracking different generations of a file is valuable, for example, if your site maintains development histories for projects. The cataloging system also requires that certain system options have particular values for cataloging to work.

The archiving subsystem tracks four backup copies of disk files, but it does not keep track of different generations of a file. It is easy to use for all basic file protection purposes.

The cataloging system and the archiving system can operate on the same system at the same time. They work independently and in parallel, but do not interfere with each other. A comparison of the two file protection systems, as well as complete descriptions of the systems, are in the *A Series Disk Subsystem Administration and Operations Guide.*

This section contains descriptions of the cataloging system and the archiving subsystem. For more extensive descriptions, see the *Disk Subsystem Guide.*

The LOCKEDFILE file attribute is also explained in this section. The LOCKEDFILE file attribute protects disk file from accidental removal, replacement, or name change. For detailed information about the LOCKEDFILE file attribute, refer to the *File Attribute Programming Reference Manual.*

Protecting Disk Files with the Cataloging System

This subsection shows you how to use the cataloging feature, and deals with the following topics:

- Normal cataloging operation
- Setting up cataloging for the first time
- Making backup copies of the catalog
- Rebuilding the catalog file
- Updating volumed tapes and volumed disks
- Replacing a damaged disk on a cataloging system
- The impact of cataloging on system performance

How Cataloging Functions

The cataloging feature provides an automated method of locating where backup copies of disk files and tape files are stored. The system handles cataloged disk files and cataloged tape files similarly, but there are differences to be aware of. These differences are discussed later in this section under "Updating the Volume Library and Volume Directory."

When a file is made permanent on a cataloging system, an entry for that file is placed in the catalog. If a permanent cataloged file is removed and there are no backup copies of the file, the catalog entry for the file is deleted. If a permanent cataloged file is removed and there are backup copies of the file, the file is marked in the catalog as being nonresident.

This section uses the term *resident* to refer to the primary copy (as opposed to a backup copy) of a file that is stored on disk, regardless of whether the disk is online or not. A file is nonresident if it is stored only on a backup tape or if it is a backup copy of a file and is stored on another disk family. Only one version of a file can be resident at one time. In this section, the terms resident and nonresident do not pertain to the file attribute RESIDENT.

Catalog Components

A cataloging system stores the access structure in the file SYSTEM/CATALOG/<family index number>. The SYSTEM/CATALOG file also stores other information that pertains to cataloging. SYSTEM/CATALOG contains three other structures: the volume library, the catalog, and an index for cataloged tape files.

The volume library is a component in the catalog file that keeps track of all volumed disks and tapes used by the system. A *volumed* disk or tape is one that has been added to the volume library with the *VOLUME ADD* WFL statement. Note that disks and tapes often are referred to as volumes, whether they have been entered into the volume library or not. The volume library contains an entry for each volumed disk or tape family. There is only one entry for a multidisk family.

The PV (Print Volume) system command returns information about how a particular disk or tape is volumed. Also, the LISTVOLUMELIB utility generates a printer listing of all the volumes in the volume library. If you are using InfoGuard on your system, you can use the SYSTEM/LISTVOLUME utility to obtain reports on the status of volumes in the volume library and/or volume directory. For more information, refer to "Using SYSTEM/LISTVOLUME" later in this section.

The system uses a special structure called the Volume Access Structure Table (VAST) to access volume library entries. The system handles the VAST automatically. The only time you are aware of the VAST is when certain system messages refer to it.

The bulk of the catalog file is made up of the catalog, which stores information about the backup copies for cataloged files. A disk file is a cataloged file if the catalog keeps track of its backup copies. Cataloged files can be stored only on a disk that has been entered into the volume library. The cataloging feature can keep track of the backup copies only if the copies are stored on a disk or tape that has been entered into the volume library.

Tracking File Generations on Cataloging Systems

You might want to keep one or two backup copies of a file. You might also want to retain older versions of a file for historical or developmental reasons. Cataloging can be used to keep track of these copies and versions automatically.

The different copies of a file are referred to as *generations*. A generation is determined by two file attributes, CYCLE and VERSION, and also the timestamp of the file.

The timestamp is a system attribute for each file that the system maintains automatically to show the last time the file was altered, including changes to its data, name, security status, and certain file attributes. If a file has never been altered, the timestamp shows the time and date the file was created.

When a new generation of a file is created and made permanent, the previous resident generation of the file is removed. If there are backup copies of the previous resident generation, an entry for that generation remains in the catalog. If there are no backup copies for the previous resident generation, its catalog entry is deleted.

When a new disk file is created, the MCP cannot check the contents of the file to see if it is a new generation of an existing cataloged file or if it is a completely different file. The MCP checks only to see that both the file name and the family name are the same. This is important to remember if you do not use CYCLE and VERSION and you copy a disk file to backup media and then remove the resident disk file. If you then create a different, unrelated file on the same family with the same name, the MCP automatically treats the new file as the newest generation of the other file that had the same file name.

Cataloging enables you to keep track of up to seven different generations of a file and up to two backup copies of each generation. The CATALOGLEVELSET define, which is compiled into the MCP, specifies the catalog level. The catalog level determines how many generations your installation can have. The catalog level is assigned the value 3 when the MCP is shipped to your installation, which means that the catalog can keep track of four generations (including the resident generation). You can recompile the MCP with CATALOGLEVELSET equal to a value in the range 1 through 7.

For more information about recompiling the MCP for cataloging, refer to "Setting Up Cataloging" later in this section. To find out what the catalog level is at your installation, use the WM (What MCP) system command.

If the maximum number of generations has been reached and a new copy is added, the generation with the *worst genealogy* is deleted from the catalog. The version with the worst genealogy is defined as the generation with the lowest CYCLE file attribute, the lowest VERSION file attribute within that CYCLE, and the least recent timestamp within that CYCLE and VERSION. The MCP option ARCHIVING allows you to save information about older generations after they are deleted from the catalog.

If the USECATALOG file attribute is TRUE when an existing disk file (a file for which the NEWFILE file attribute is equal to FALSE) is accessed, the system locates the file through the catalog. By default, the system accesses the generation with the *best genealogy* unless you specify a particular CYCLE and VERSION or use the GENERATION file attribute. The version with the best genealogy is defined as the generation with the highest CYCLE file attribute, the highest VERSION file attribute within that CYCLE, and the most recent timestamp within that CYCLE and VERSION. The GENERATION file attribute allows you to access a particular generation by specifying an integer value that designates that generation. The value of GENERATION can range from 0 for the generation with the best genealogy to the value of CATALOGLEVELSET minus 1 for the generation with the worst genealogy. The default value of GENERATION is 0. GENERATION is ignored if USECATALOG is FALSE when the file is accessed.

The CYCLE and VERSION file attributes function in the following manner on existing disk files if USECATALOG is TRUE and if GENERATION has not been specified or has been assigned the value 0:

- If you specify CYCLE and VERSION when you want to access the file, the system locates the generation that has the exact CYCLE and VERSION values you specify. If you do not assign a value to VERSION, the system locates the generation with the CYCLE value you specify and the VERSION equal to 0. The following situations can occur when the system tries to locate the proper generation:
  - If there is no entry in the catalog for a file with the file name you specify, a NO FILE message is displayed on the ODT.
  - If there is more than one generation of the file with the specified CYCLE and VERSION values, the system chooses the generation with the most recent timestamp. If that generation is resident, the system accesses the file. If that generation is not resident, the system displays a NO FILE message on the ODT. This message also displays the serial number of the backup disk on which that generation is stored.
- If you do not specify CYCLE and VERSION when you want to access a file, the system locates the generation that has the highest CYCLE, the highest VERSION within that CYCLE, and the most recent timestamp within that CYCLE and VERSION. The following situations can occur when the system tries to locate the proper generation:
  - If there is no entry in the catalog for a file with the file name you specify, a NO FILE message is displayed on the ODT.
  - If there is an entry in the catalog for the file, and the generation with the best genealogy is resident, the system accesses the file.
  - If there is an entry in the catalog for the file but the generation with the best genealogy is not resident, a NO FILE message is displayed on the ODT. This message also displays the serial numbers of the disks on which backup copies of the generation are stored.

CYCLE and VERSION function in the following manner on existing files if USECATALOG is TRUE, and you assign a value greater than 0 to the GENERATION file attribute.

- If you specify CYCLE and VERSION when you want to access the file, the system first locates the generations that have the CYCLE and VERSION values you specify. If you do not assign a value to VERSION, the system locates the generations with the CYCLE value you specify and the VERSION equal to 0. The system then compares the timestamps of the generations within the selected CYCLE and VERSION and chooses the generation based on the value of GENERATION.

  For example, if you assign GENERATION the value of 0, the system selects the most recent timestamp within the appropriate CYCLE and VERSION. If you assign GENERATION the value of 1, the system selects the next most recent timestamp within the appropriate CYCLE and VERSION, and so on.

  The following situations can occur when the system tries to locate the proper generation:

  - If there is no entry in the catalog for a file with the file name you specify, a NO FILE message is displayed on the ODT.
  - If the system locates generations with the correct CYCLE and VERSION, but the generation specified by the GENERATION file attribute is missing from the catalog, an UNMATCHED GENEALOGY message is displayed on the ODT. The specified generation is missing if GENERATION is assigned a value greater than or equal to the number of generations within that CYCLE and VERSION. For example, if there are three generations within that CYCLE and VERSION (generation 0, generation 1, and generation 2) and you assign GENERATION the value of 3, the system cannot find the correct generation.
  - If the system locates the correct generation based on CYCLE, VERSION, and GENERATION, but the specified generation is not resident, a NO FILE message is displayed on the ODT. This message also displays the serial number of the backup disk on which that generation is stored.

- If you do not specify CYCLE and VERSION when you want to access the file, the system ignores the values of CYCLE and VERSION and uses only the timestamps to rank the generations. The most recent timestamp becomes generation 0, the next most recent timestamp becomes generation 1, and so on. The system then chooses the generation specified by the value of the GENERATION file attribute.

  The following situations can occur when the system tries to locate the proper generation:

  - If there is no entry in the catalog for a file with the file name you specify, a NO FILE message is displayed on the ODT.
  - If the specified generation is not resident, a NO FILE message is displayed on the ODT. This message also displays the serial number of the backup disk on which that generation is stored.
  - If there are not enough generations of the file in the catalog (for instance, if there are only two generations but GENERATION is assigned the value 3), an UNMATCHED GENEALOGY message is displayed on the ODT.
  - If the specified generation is resident, the system accesses the file.

Tracking File Generations on Noncataloging Systems

In certain application programs, it is convenient to create different generations of the same file, such as a new generation for each time the program runs. There are two ways to differentiate the generations of a file on a noncataloging system.

The first approach is to assign a different name to each generation. An example of this approach would be PAYABLE/001, PAYABLE/002, PAYABLE/003, and so on. All the generations can be online at once because they have different file names.

The second approach is to use the CYCLE and VERSION file attributes to distinguish each generation. CYCLE and VERSION are integer values; the higher the value of CYCLE and the higher the value of VERSION within each CYCLE, the better the genealogy of the generation is said to be relative to other generations. Only one generation can be online simultaneously because the file names are the same. If explicit values are not assigned to CYCLE and VERSION, the default value is 1 for CYCLE and 0 for VERSION. You must assign CYCLE to use VERSION, but assigning VERSION is optional when you use CYCLE.

The following example illustrates the use of CYCLE and VERSION values on a noncataloging system. In this example, the first entry is the generation with the best genealogy and the last entry is the generation with the worst genealogy.

CYCLE	VERSION
4	0
3	3
3	2
2	3
2	2
2	1
2	0
1	5
1	0

CYCLE and VERSION function in the following manner when you try to access an existing file on noncataloging systems:

- If you specify CYCLE and VERSION when you want to access a file, the system locates the generation that has the CYCLE and VERSION values you specify. If a file with the file name you specify is not online or does not exist, a NO FILE message is displayed on the ODT. If a file with the specified file name but the wrong CYCLE and VERSION is online, the *UNMATCHED GENEALOGY* message is displayed on the ODT.
- If you specify CYCLE, but not VERSION, when you want to access a file, the system locates the generation with the specified CYCLE value and the VERSION equal to 0. If the generation that is online does not have that CYCLE and VERSION, the *UNMATCHED GENEALOGY* message is displayed on the ODT. If no generation of the file is online, the NO FILE message is displayed on the ODT.

- If you do not specify CYCLE and VERSION when you want to access the file, the system locates the generation that is online. If no generation of the file is online, a NO FILE message is displayed on the ODT.

Note that if the AUTORM system option is set, then the system automatically removes the old file with the duplicated name from the family.

Setting Up Cataloging

The MCP is supplied with a default catalog level of 3, meaning that if the MCP option CATALOGING is enabled, the catalog can keep track of four generations of files including the resident generation.

Use the WM (What MCP) system command to determine whether cataloging is already active. If the value displayed for CATALOG LEVEL is greater than 0 (zero), cataloging is already active. The value displayed indicates the number of generations (including the resident generation) that the system can currently track.

If you want to keep track of a different number of generations and your site is licensed for source code, then before you enable the CATALOGING option you must recompile the MCP as follows:

1. Look in the MCP source file for the following record:

   ```
   CATALOGLEVELSET = 3#
   ```

   The MCP source file has a name similar to *SYMBOL/MCP.
2. Use the Editor or SYSTEM/PATCH to change the 3 to a number in the range 1 through 7, depending on how many file generations you want to track besides the current generation.
3. Save the new MCP source file under a different name and compile it using the NEWP compiler or the WFL job WFL/COMPILE/SOFTWARE, which is distributed with the system.
4. Halt/load the system, ensuring that it loads the newly compiled version of the MCP.

For more information about compiling Unisys system software and using the various compile-time options, refer to the *Software Release Installation Guide.*

To set up cataloging for the first time, perform the following steps.

Note: *Do not use this procedure if you are already running on a cataloging system, because the current catalog becomes inaccessible as a result.*

1. Use the DL (Disk Location) system command to designate a disk family as the catalog family.
2. Use the OP + CATALOGING version of the OP (Options) system command to enable the MCP option CATALOGING.

3. Halt/load the system. The system then displays the following message on the ODT:

   ```
   OK TO CREATE NEW CATALOG
   ```

4. Enter the reply *OK*. The system then creates the new SYSTEM/CATALOG file on the catalog family.
5. Use the WFL statement *VOLUME ADD* to enter into the volume library each disk and tape on which you want to store cataloged files. It is suggested that all disks and tapes used on your system be added to the volume library.
6. Use the WFL statement *CATALOG ADD* to enter existing files into the catalog.
7. If you want files to be cataloged by default, use the OP + USECATDEFAULT version of the OP (Options) system command to assign TRUE as the default value of the USECATALOG file attribute.

The WFL *RUN* statement acts only on the generation with the best genealogy if the OP USECATDEFAULT system option is enabled. If the OP USECATDEFAULT system option is not enabled, the RUN statement acts on the resident generation.

If you use the RUN statement and the generation with the best genealogy is not resident, the system displays a NO FILE message on the ODT. This message also displays the serial numbers of the media on which backup copies of the file are stored.

The CS (Change Supervisor) system command acts only on the generation with the best genealogy if the OP USECATDEFAULT system option is set.

The CM, MC (Make Compiler), SI, and SL system commands always act on the resident generation, regardless of whether it has the best genealogy and regardless of whether the OP USECATDEFAULT system option is enabled.

Using Cataloging

The following procedures are used when operating a cataloging system:

- Entering files into the catalog
- Accessing cataloged files
- Deleting catalog entries
- Purging catalog backup tapes

Entering Files into the Catalog

Once cataloging is set up on your system, you enter a file into the catalog file by performing one of the following actions. The file must be stored on a volumed disk—a disk for which a WFL *VOLUME ADD* statement has been executed (as described previously under "Setting Up Cataloging.")

- Assign the USECATALOG file attribute the value TRUE before creating the file. The file will be cataloged automatically when the file is made permanent.
- Operate the system with the system option USECATDEFAULT enabled. This makes the default value of the USECATALOG file attribute TRUE. All permanent files are entered into the catalog when they are created unless their USECATALOG file attribute is assigned the value FALSE.
- Use the WFL *CATALOG ADD* statement to mark a permanent file as cataloged.
- Use either the WFL statement *COPY & CATALOG* or *ADD & CATALOG* to copy the file to disk from backup media and enter the file into the catalog. The copied version is marked in the catalog as the resident version of the file and the source version is marked in the catalog as a backup copy.
- Use the SM (Send to MCS or Database) system command to set the CANDE *CATDEFAULT* option. Enter the following when the CANDE MCS is running:

  ```
  <mix #> SM OP + CATDEFAULT
  ```

  To find the correct mix number, enter *?A* in CANDE, or enter *A* in the Action field of any MARC screen. The system displays the list of active entries. The desired number is the one that appears under "Mix" for the JOB *SYSTEM/CANDE entry.

  When CANDE work files are saved, the CANDE *CATDEFAULT* option governs whether those files become cataloged files.

Using the Cataloging Subsystem to Backup and Restore Disk Files

The normal operation of the catalog backup facility consists of making backup copies of disk files and, when required, restoring some of those backups to disk. These operations are briefly described here.

To make backup copies of disk files, use the WFL *COPY & BACKUP* statement. The files to be copied must have already been marked as cataloged files, and the output tape volume must have an entry in the volume library.

To restore a backup copy of a file to disk, proceed as follows:

1. Use the PD system command or the SYSTEM/FILEDATA CATALOGINFO function to display or print the catalog information. Doing so gives you the serial number of tapes containing backup copies of the file. For example:

   ```
   PD (UX)F1 ON XPACK
   ```

   PD and SYSTEM/FILEDATA report the backup information even if no resident version of the file exists on disk at the time.
2. Load one of the tapes on a tape drive, and issue a PER MT system command to determine the name of the tape.

3. Issue a WFL *COPY* statement that gives the name of the file, the name (and serial number) of the tape, and the name of the disk family. For example:

```
COPY & COMPARE (UX)F1 FROM UXT66 (SERIALNO=2537)
  TO XPACK (PACK)
```

A *NO FILE* RSVP message for a cataloged file that has a backup includes the serial number of the backup tape on which the file can be found.

Refer to the *WFL Reference Manual* for more information on the COPY & BACKUP command.

Making Backup Copies of the Catalog Directory

You might want to make backup copies of the catalog directory occasionally to provide backups in case any catalog data on disk becomes corrupted or the catalog family fails. The needs of your installation determine how often you should copy the SYSTEM/CATALOG file. A backup catalog does not contain any information that has been added to the catalog since that backup copy was made. If the catalog entries are updated frequently and recovering the cataloging information is critical on your system, you will want to back up the catalog frequently.

The COPYCAT (Copy Catalog) system command makes an inactive copy on disk of the currently active system catalog file. When you have made the copy, you can transfer it to tape and store it offline as a safety measure. Do not copy the catalog directly to tape because some catalog records might be changed during the copy process. The COPYCAT command stops all catalog updates while the backup copy is made. The latest catalog copy can be copied back in after a catastrophic failure of the catalog file or the catalog family. You can enter the COPYCAT command in CANDE, or you can use the CPYCAT selection on the MARC *DSM* menu, which is accessible from the *FILE* menu.

The system also can have duplicate online copies of the catalog; refer to "Duplicating the Catalog File" in Section 5.

When you restore the catalog from a backup copy, some catalog information is not up-to-date. The names and serial numbers of backup media are not available for files that have been copied with a system command such as COPY & BACKUP since the last time the catalog was backed up. As a result, the backup copies of files that were made after the catalog was copied to tape cannot be accessed through the catalog. Also, the names and other information are not available for volumed tapes that have changed since the catalog was backed up.

When you need to replace the current catalog, you can perform one of the following operations:

- If the catalog family is usable, you can replace the current catalog so that the backup catalog is on the same family and is thus easy for you to locate. Replacing the current catalog requires extra steps, however, because the current catalog is marked as a nonremovable system file.
- If the catalog family cannot be restored, you must designate a new catalog family and copy the backup catalog to that family.

The procedure for replacing the catalog appears in the *Disk Subsystem Guide.*

Accessing Cataloged Files

The catalog file can keep track of up to seven generations of each cataloged file, depending on the catalog level. You can choose the generation of the file that the system accesses. The system uses the catalog to access the file if the value of the USECATALOG file attribute is TRUE or if the USECATDEFAULT system option is enabled so that the default value of USECATALOG is TRUE. USECATALOG takes precedence over USECATDEFAULT. If USECATALOG is assigned the value TRUE, but the default value of USECATALOG is FALSE because USECATDEFAULT is not enabled, then USECATALOG still has the value TRUE. If USECATALOG is assigned the value FALSE, but the default value of USECATALOG is TRUE because USECATDEFAULT is enabled, then the value of USECATALOG is still FALSE.

If USECATALOG is TRUE, the system accesses the generation with the best genealogy unless you request another generation by using the CYCLE, VERSION, or GENERATION file attributes.

USECATALOG must be TRUE in order for the system to automatically access the generation with the best genealogy. If USECATALOG is FALSE, the system accesses the resident version of the file by default. If the generation with the best genealogy has been removed and another generation has been copied to disk, and thus is the resident version, the system might be using an outdated version of the file.

The WFL statements *REMOVE, CHANGE, ADD, SECURITY,* and *CATALOG ADD* always act on the resident generation of the file, regardless of whether it has the best genealogy and regardless of whether USECATDEFAULT is enabled.

When you use the WFL statement *CATALOG DELETE,* you can specify a particular generation by assigning CYCLE, VERSION, or GENERATION. If you do not specify a particular generation, the *CATALOG DELETE* command acts on the generation with the best genealogy. The WFL statement *CATALOG PURGE* acts on all the generations of the file. Refer to "Removing Catalog Entries" later in this section for more information about the WFL statements *CATALOG DELETE* and *CATALOG PURGE.*

To examine backup file information such as the available generations, use the PD (Print Directory) system command. The display generated by the PD command refers to the generations as entries. These entries range from 1 for the generation with the best genealogy to the value of CATALOGLEVELSET for the generation with the worst genealogy. Note that this numbering is different from that used in the GENERATION file attribute. If you use the PD command is used to examine the resident version and it is not the generation with the best genealogy, some of the PD display is omitted. The FILEDATA utility also can be used to display information about the file attributes and backup copies of a file.

Once a file has been entered into the catalog and backed up, it remains in the catalog even if the resident version is removed. You might want to remove the resident version because you need to make disk space available, because the resident version is damaged, or because you want to create a new version of the file.

If you try to open a file that has been removed, the system displays a NO FILE message on the ODT. This message also displays the serial numbers of the media on which backup copies of the file are stored. The system operator should locate a backup of the file and copy it back in with the WFL *COPY* statement.

Deleting Catalog Entries

Information about a file can be removed from the catalog in two ways:

- The WFL statement *CATALOG DELETE* removes references to a particular generation, including the one that is resident.
- The WFL statement *CATALOG PURGE* removes all the backup information for a file.

The WFL statements *CATALOG DELETE* and *CATALOG PURGE* delete only catalog entries; the resident and backup files are still available, but cannot be accessed through the catalog. If the copy specified in the WFL statement *CATALOG DELETE* is the resident version, the copy still remains the resident version after its catalog entry has been deleted.

Purging Catalog Backup Tapes

After using cataloging for some time, you might have backup copies on tape that are no longer needed. You can purge these backup tapes and use them for other purposes. Purging the backup tapes causes the system to automatically change the tape names in the volume library to SCRATCH. However, the catalog entries for the files that were backed up on each tape will still indicate that backup copies are stored on that tape and refer to its old name.

Using Tape Security

The volume directory is a data structure in the tape security subsystem that stores information about tape volumes at your installation by tape serial number. The volume directory includes the following information for each tape volume family: current volume name, tape volume ownership, and tape file security attributes. The volume directory is used by the system when a tape volume is readied or purged and whenever a tape file is opened for input or output. The tape volume directory is a feature that is only available with InfoGuard security-enhancement software.

The tape volume directory has the following two sections:

- The data section contains data records that describe individual tape volume families.
- The key section contains a structure used by the system to rapidly locate individual records in the data section.

The system frequently changes volume directory entries to track the changing status of the tape volumes. An operator can issue commands to rebuild the volume directory if it becomes corrupted. If you designate the SECOPT TAPECHECK option equal to

NONE, then the system does not create, reference, or update the volume directory data structure. To have volume directory capability, you must designate the SECOPT TAPECHECK option equal to AUTOMATIC as follows:

```
SECOPT TAPECHECK=AUTOMATIC
```

When you enter the command shown above, it does not become effective until the next time that you halt/load the system.

Rebuilding the Catalog

When a cataloging system needs to perform a family rebuild to recover from directory errors, it first rebuilds the family and then rebuilds the catalog. During the rebuilding of a family on a cataloging system, the system reads the flat directory to determine which files are resident and inserts the names of the resident files into the catalog and the family access structure table (FAST).

Rebuilding the catalog is a lengthy process, during which the system reads the catalog to extract all the backup information for each cataloged file and enters the names of the files that have backup copies into the FAST.

The system then rebuilds the family.

For more information about the FAST and rebuilding families, refer to "Rebuilding Families" in Section 5.

Updating the Volume Library and Volume Directory

Cataloging systems and tape security subsystems handle disk and tape volumes differently in terms of how they are added to, updated in, or deleted from the volume library or volume directory.

Updating Volumed Tapes

The system automatically updates the volume library and/or volume directory entry for a volumed tape when you purge the tape with the PG (Purge) system command. The system also automatically updates the volume library and/or volume directory entry for a tape when you place a different file on a volumed tape. The automatic updating of the volume library or volume directory entry does not take place if any of the following occur:

- The tape is purged while cataloging or the tape security subsystem is not in use.
- The tape is purged on a different system.
- A new file is stored on the tape while cataloging or the tape security subsystem is not in use.
- A new file is stored on the tape while the tape is being used on another system.

Updating Volumed Disks

Volumed disks are handled differently. If you want to rename a disk or disk family, use the RC (Reconfigure Disk) or LB (Relabel Pack) system command. When you want to purge the files from a disk or disk family, use the PG (Purge) system command. However, when you use these commands on a volumed disk, first use the WFL statement *VOLUME DELETE* to remove the disk from the volume library. After you reconfigure, rename, or purge the disk, you can reenter it into the volume library with the WFL statement *VOLUME ADD*.

To add a multidisk family to a volume library, use the name of the family and the serial number of the disk unit with the family index number 1. Do not add the other units in the family to the volume library, even if the unit bearing family index number 1 is not currently online.

Using SYSTEM/LISTVOLUME

SYSTEM/LISTVOLUME is a utility in InfoGuard that reports on the status of volumes in the volume library, the volume directory, or both. To use this utility to report on the volume library, your installation must run with the MCP option OP CATALOGING. (For instructions on setting MCP options, see "Choosing Additional MCP Options" in Section 1.) To use the utility to report on the volume directory, your installation must run with the InfoGuard option. You must first enter the system command

```
SECOPT TAPECHECK = AUTOMATIC
```

Only privileged users can use this utility. Depending on the size of the volume library or volume disk, a large amount of disk space might be required by the program.

Replacing a Damaged Volumed Disk

Damage to the base unit for a volumed family can render all files on the family inaccessible. To preserve the backup information for the files that were on the original family, you must substitute another base unit for the damaged one or repair and reuse the damaged disk.

The steps for substituting a new base unit for the damaged one so that the family inherits the backup information is described in the *Disk Subsystem Guide*.

Impact of Cataloging on System Performance

Using cataloging to automate the locating of backup copies of files can affect system performance. Each time a disk file is opened or closed on a noncataloging system, two to three disk I/O operations occur. One to two additional I/O operations are required to open or close a disk file on a cataloging system.

Rebuilds take substantially longer on a cataloging system than on a noncataloging system because both a catalog rebuild and a family rebuild must be done. Rebuilds do occur less frequently on a cataloging system, though, because the catalog saves an up-to-date

description of the files on a disk. If the disk is brought online and none of the files have been altered on another system, no rebuild is necessary.

Another tradeoff involved in cataloging is that the SYSTEM/CATALOG file is usually large. At least one sector and sometimes several sectors are needed to store the catalog information for each disk file on the system. Unisys recommends that the SYSTEM/CATALOG file be stored on a family with files that are not accessed frequently. Frequent I/O operations to the family containing SYSTEM/CATALOG slow the handling of the catalog and thus the performance of the entire system.

Protecting Disk Files with the Archiving Subsystem

This subsection describes the archiving subsystem from an administrative point of view. The functions and features of the archiving subsystem pertaining to system administration are described; note that this information is not intended to describe "how-to" procedures. Refer to the *Disk Subsystem Guide* for detailed information about the subsystem.

This subsection covers the following information:

- A description of the archiving subsystem
- Administration considerations for archiving
- Setting up archiving for the first time
- Making backup copies of archive directories
- Using the archiving subsystem to backup and restore disk files

Overview of the Archiving Subsystem

The archiving subsystem consists of three main components. These components are

- Library maintenance operations. These operations are performed through archiving statements which can be included in WFL jobs or entered at the ODT. These library maintenance operations include copying and transferring disk files to backup tapes, restoring backup tape files to disk, and merging backup tape files from many tapes to a single tape or tape set.
- Archive directories. The archiving subsystem creates and maintains a directory on each on-line disk family where archive operations have been performed. The archive directory contains the names, locations, and attributes of disk files that have been transferred through archive operations to tape, or merged from many tapes to a single tape or tape set. These directories reside on the DL catalog family.
- The archiving support library. The support library is used by the archiving subsystem to control which files are copied during an archive operation. The library selects or rejects files presented to it by the archiving subsystem for archiving operations. The library makes these selections based on various selection criteria, and is sometimes referred to as the *selector library*. The standard symbolic file for the support library is named *SYMBOL/ARCHIVESUPPORT; the standard code file for the support library is named *SYSTEM/ARCHIVESUPPORT.

The following features provided by the archiving subsystem are of interest to the system administrator. Note that this is not a complete list of features. The archiving subsystem allows you to do the following:

- Back up disk files on a disk family with the WFL *ARCHIVE FULL* statement.
- Back up only those disk files that have been changed or added since the last archive backup operation with a WFL *ARCHIVE DIFFERENTIAL* or *ARCHIVE INCREMENTAL* statement.
- Automatically back up and remove disk files to free up disk space with a WFL *ARCHIVE ROLLOUT* statement.
- Merge previously backed up tape files from many tapes to one tape or tape set with the *ARCHIVE MERGE* statement.
- Restore backup copies of files to disk with a WFL *ARCHIVE RESTORE* or *ARCHIVE RESTORE/ADD* statement.
- Automatically recover an archived file from tape when needed by a user or a program with the AUTORESTORE feature.

The archiving subsystem automatically tracks the location of archived files through the archive directory. The subsystem tracks the four most recent backup copies of each file, but the archive directory tracks backup copies of only one generation of a file.

For a complete description of the functions and features of the archiving subsystem, see the *Disk Subsystem Guide.*

Setting Up Archiving

Several actions must be taken before archiving can be activated on your operating system:

1. Initiate your operating system.

   When your operating system is first initialized, the system allocates archive directories on the DL CATALOG disk family. The system allocates a separate archive directory for each online family. Each of these directories initially requires 600 disk segments on the DL CATALOG family. When the first backup entries are recorded in an archive directory, that directory is immediately expanded by another 600 disk segments.

2. Assign the ARCHIVESUPPORT function to its support library. You can do so by issuing the system command

   ```
   SL ARCHIVESUPPORT=SYSTEM/ARCHIVESUPPORT
   ```

   The SYSTEM/ARCHIVESUPPORT library is used by all the WFL *ARCHIVE* statements except *ARCHIVE DELETE.*.

3. Set the AUTORESTORE system option to YES, DONTCARE, or NEVER. Refer to the *System Commands Reference Manual* for more information about the AUTORESTORE system option.

Making Backup Copies of Archive Directories

You might want to make backup copies of the archive file itself occasionally to provide backups in case any data on disk becomes corrupted or is accidentally removed. The needs of your installation determine how often you should copy the archive directories. A backup copy of an archive directory contains no information that was added to the archive files since that backup copy was created.

To make a backup copy of the archive directory for a disk family, use the ARCCOPY system command. The system prompts you for the next steps to perform. Refer to the *System Commands Reference Manual* for more complete information on the ARCCOPY system command.

Deleting Archive Entries

You can delete information from an archive directory in one of two ways:

- By using a WFL *ARCHIVE PURGE* statement
- By calling the intrinsic DCALGOL SETSTATUS procedure with the values TYPE = 3 and SUBTYPE = 1

Using the Archiving Subsystem to Backup and Restore Disk Files

The normal operation of the archive subsystem consists of making backup copies of disk files and, when required, restoring some of those backups to disk. These operations are briefly described here.

To make backup copies of disk files you can use WFL *ARCHIVE FULL*, *ARCHIVE INCREMENTAL*, *ARCHIVE DIFFERENTIAL*, and *ARCHIVE ROLLOUT* statements. These statements select and copy files from a disk family to one or more backup tapes.

Unisys provides three ways to restore or copy backup copies of archived files from tape:

- Use the WFL *ARCHIVE* statements.
- Use the system AUTORESTORE option.
- Use ordinary library maintenance WFL *COPY* statements.

To restore a backup copy of a file to disk you can simply use the WFL *ARCHIVE RESTORE* or *ARCHIVE RESTOREADD* statements. These processes look up the names and serial numbers of the backup tapes containing the files to be restored. Library maintenance then requests these tapes one by one. When you load a requested tape on a tape drive, library maintenance copies files from it to disk.

When a "NO FILE" RSVP message is issued for a file that has an archive backup copy the message will include the name and serial number of the backup tape on which the file can be found. If the AUTORESTORE system option is YES and if the process requesting the file is the owner of the file (as determined by usercode), then the system automatically starts an archive restore process. When you load the requested tape on the tape drive, library maintenance copies the missing file to disk.

You can also use ordinary library maintenance to make a copy of a nonresident file directly from a backup tape to another tape or to a disk family other than its original family. To do so, proceed as follows:

1. Use the PD system command or the SYSTEM/FILEDATA ARCHIVEINFO function to display or print the archive information. Doing so gives you the names and serial numbers of tapes containing backup copies of the file. For example:

   ```
   PD (UX)F1 ON XPACK
   ```

   PD and SYSTEM/FILEDATA report the backup information even if no resident version of the file exists on disk at the time.
2. Load one of the tapes on a tape drive and ready it.
3. Issue a WFL *COPY* statement that gives the name of the file, the name (and serial number) of the tape, and the destination. For example:

   ```
   COPY & COMPARE (UX)F1 FROM UXT66 (SERIALNO=2537)
      TO WORKPACK (PACK);
   ```

Using the LOCKEDFILE File Attribute

Use the LOCKEDFILE attribute to determine or specify that a permanent disk file not be removed, replaced, or have its name changed.

The following conditions occur when LOCKEDFILE has a value of TRUE:

- A program cannot close a disk file with the PURGE option until the value of LOCKEDFILE has been changed to FALSE by a privileged user or the owner of the file.
- The WFL *REMOVE* and *CHANGE* commands have no effect on disk files that have LOCKEDFILE set to TRUE.
- Programs cannot replace a disk file that has LOCKEDFILE set to TRUE.

When a logical file is assigned to a physical file and the LOCKEDFILE attribute is interrogated the attribute value of the physical file is displayed.

Disk Files

An attempt to replace a disk file that has a LOCKEDFILE value of TRUE causes the following message to be displayed:

```
DUP FILE <filename> (LOCKEDFILE)
```

If the file must be removed, a privileged user or the owner of the file must use the WFL *ALTER* statement to change the value of the LOCKEDFILE attribute to FALSE, and then enter the *<mix> OK* command in response to the message. If the *<mix> OF* command is entered, the job continues without replacing the locked file and a close error is returned to the program.

When the LOCKEDFILE value is changed for a logical file that is assigned to a disk file, the value for the permanent disk file is also changed.

Section 7
Understanding the Print System

The A Series Print System, comprising the Print System and Remote Print System software, provides controls for on-site (local), data comm (remote), network, and virtual printing devices. Some Print System commands are used to establish a system-wide printing environment; other Print System commands configure individual devices. Although not required, establishing criteria for the use of printing resources can help your site run more efficiently.

All A Series systems come with the PrintS software, which controls local and virtual devices. ReprintS is optional, but is needed if your site is going to use data comm devices or if you are going to route print requests to printers on another system connected through a network supported by Unisys. However, once ReprintS is installed and initialized, remote printers are controlled with the same commands used for local printers.

All Print System (PS) commands can be entered from an ODT or through the MARC interface (either on the MARC Action line or by accessing the MARC *PS* menu). Privileged users, and MARC users with SYSTEMUSER status, have access to all PS commands through MARC; nonprivileged users have access to only a subset of the PS commands, and in general can only initiate actions that affect their own print requests.

In addition, you can define one or more Print System control stations and control users for a particular device. If an output device has a control station, PS commands affecting that device can be entered only from that station. If a device has a control user, PS commands affecting the device can be entered only under that usercode. Refer to "Control Stations and Users" in Section 8 for more information.

There are five system files associated with the Print System. If you have installed your software using the Simple Installation (SI) program, you should verify that the following three files have been copied to the designated disk family from the release tape. Otherwise, copy these files to a disk family from the release tape.

SYSTEM/PRINT/SUPPORT	Provides library functions
SYSTEM/PRINT/ROUTER	Routes print requests to devices
SYSTEM/PRINT/BACKUP/PROCESSOR	Executes print requests

These three files can be located on any disk family, but all three must be placed on the same family. Refer to Section 4, "Allocating System Files," for more information about how to allocate system files.

Additionally, the Print System uses two files to maintain information about print requests and output device configuration:

SYSTEM/PRINTERINFO	Maintains device configuration information
SYSTEM/BACKUPFILELIST	Stores print request and print server information

These files are created when the Print System is initialized and placed on the family specified by the DL BACKUP command. The DL BACKUP family can be any recognized family on the system.

If you have installed new software or an update, you should verify that the version levels are compatible. You can display the Print System software version number by entering the PS command from the MARC Action line or at an ODT. Be sure that the version number displayed is compatible with the current version of the MCP and that a single version number is returned. Do not use different versions of the various components of PrintS software.

Concepts and Terminology

Before considering particular administrative decisions, you should become familiar with the concepts and terminology used when discussing the A Series Print System. Most Print System features apply to backup files spooled to disk. These files are printed automatically unless a user explicitly specifies otherwise.

Spooling (Indirect Output)

The A Series Print System uses a process called *spooling*, which causes program output to be written to a disk or tape file rather than directly to a printer. This process allows a site's printing devices to operate at their maximum capacity because spooled files can be queued and then printed continuously as each current print job is completed.

The use of spooling is referred to as an indirect output mode or environment. This term is used to contrast the method with a direct output mode, which assigns a printer to an executing program.

Spooling of printer and punch output is controlled by the LPBDONLY and CPBDONLY system options. With these options, spooling can be in effect for printer files only, for card punch files only, for both, or for neither (which would mean that the system is in a direct output mode). These options are set by using the OP (Options) system command. Note that a MARC *SYSTEMUSER* or a privileged user can enable and disable the spooling options by selecting the DVCCTL option from the MARC *PS* menu.

If the LPBDONLY option is set, all program output files whose KIND attribute value is PRINTER are spooled. If the CPBDONLY option is set, all program output files whose KIND attribute value is PUNCH are spooled. The disk family where spooled files are written is controlled by the DL BACKUP command.

Direct Output

If you decide not to use spooling, printer output files are routed directly to the first available output device, not to a backup medium. This process is called direct output.

If you specify direct output for a file, printing takes place while your program is executing; no print request is constructed and so a direct output file is invisible to the Print System.

Direct output links a printer to a running program for the duration of program execution. Direct output is seldom preferable; one situation where it can be useful is if you want to synchronize the serial number on a preprinted form with program output. However, running in direct output mode has several disadvantages:

- A program might have to wait until a suitable printer becomes available.
- The printer will be idle while the program performs processing not related to output.
- The printer is dedicated to the program and is unusable by other programs until the using program has completed.
- The program run time is increased because execution is slowed by the lower speed of the printer.
- You cannot use many of the features that customize printed output.

Direct output can be specified for individual jobs, tasks, or files even when the system as a whole is set up for spooling. The PRINTDISPOSITION file attribute allows the specification of direct output.

For information about how a printer responds to various error conditions (including parity errors), refer to the documentation for the printer.

Printer Backup Files

Output files that are spooled to disk or tape are referred to as printer backup files. A printer backup file is a (temporary) copy of a program output file. In addition to the data to be printed, printer backup files contain all the carriage control information specified by the program so that the final output can be properly formatted. Printer backup files also contain the information needed by PrintS to properly print the file.

Normally, printer backup files are written to a disk family where they reside until they have been printed; if they print without incident, they are removed. This handling of printer backup files can be altered in several ways through the use of task and file attributes and PS DEFAULT commands.

For instance, printer backup files can be saved after they are printed, they can be saved on disk or tape without being printed, and they can be renamed.

Storing a Printer Backup File on Disk without Printing It

By assigning the value DONTPRINT to the PRINTDISPOSITION attribute of a printer backup file, you indicate that you do not want the file printed and removed from the system. The Print System does not create a print request for such files, and thus, they are not entered into the SYSTEM/BACKUPFILELIST file.

Writing Printer Backup Files to Tape

Tape backup files are not printed automatically because the Print System does not keep a record of printer backup files written to tape in the SYSTEM/BACKUPFILELIST file. Once stored on tape, a backup file can be

- Printed directly from tape using the PB (Print Backup) system command
- Printed directly from tape using the SYSTEM/BACKUP utility
- Copied from tape to disk, then printed or used otherwise

Refer to the PS ADDFILES command, the WFL *PRINT* statement, and the PB system command in the *System Operations Guide*.

Print Requests

A print request consists of one or more backup files to be printed, along with the attributes that describe how to print those files. The Print System combines all backup files having similar characteristics from a task, job, or session, into print requests.

Printer backup files are grouped into different print requests if they have incompatible characteristics (usually the values of task and file attributes); for example, different destinations, different forms requirements, or different times to be printed. If, for instance, you specify that certain output files are to be printed at the end of the task in which they are created (EOT) and other output files at the end of the job (EOJ), then the files whose PRINTDISPOSITION attribute is assigned the value EOT are placed in a different print request from the ones to be printed at EOJ.

If the content of a file is changed between the time it is evaluated and its servicing, those changes are reflected in the print request. If, in that same interval, any attributes are changed, a conflict may arise between it and the other files in the request. If the Print System determines that a file in a request is incompatible with the assigned device, it splits the request. That is, it prints all the files it can, then regroups those it cannot print into another print request and puts that request into the print queue.

If an entire directory is to be printed, the Print System expands the directory name into a complete list of files when it constructs the print request. Files that are added to the directory after the creation of a print request are not included in that request. Files that are deleted from a directory after the print request is created but before it is serviced are not printed.

Processing a Request

After being constructed, print requests are placed in a queue for printing. Depending on the selection criteria, the request is printed when a device becomes available that meets the characteristics of the print request; for example, the correct form is mounted or a specified time has been reached.

If a backup file has been printed without incident, it is removed from the disk. However, if an error prevents a backup file from being printed completely, the request is marked as

an exception and replaced in the print queue. An exception request stays in the queue until a user intervenes to retry the request or delete it.

Once a print request is in the print queue, the request can be controlled and modified by the various Print System (PS) system commands, such as PS MODIFY, PS FORCE, and PS DELETE. For information about all Print System commands, refer to the *Print System Guide.*

Routing Print Requests

The Print System creates and routes print requests automatically. If spooling is in effect, output files are printed or punched when a job or a MARC or CANDE session ends or is split.

If you do not use task and file attributes to modify a print request when a program is run, the request is printed according to the default values. However, even if a print request is generated automatically, you can still control where, when, and how it is printed. For example, you can specify the device to which backup files are to be routed or you can delay printing until a certain time.

Task and File Attributes

Task and file attributes are control parameters that allow information to be specified about a task (program) or a file. Many task and file attributes affect printed output and can give you a great deal of control over the backup files generated by a job.

Using Task and File Attributes

Some features affected by task and file attributes are harmless in terms of controlling your system, such as assigning a new title to a file or suppressing a job summary printout. Other attributes can affect your site operations, so you might want to consider establishing policies for use of some attributes.

Another consideration concerning file attributes involves program development. File attribute assignments rarely should be assigned at run time in a program because if any of these values must be changed later on, the program code must be modified and recompiled. If file attribute values are assigned through file equation in the task initiation statement or in a file declaration, values can be changed easily whenever the program is run.

All task and file attributes have default values that are used if no other values are specified. If some default values do not suit your needs, you can create WFL jobs for running programs that use nondefault attribute values.

Precedence of File Attributes and Print Modifiers

File attribute values can be specified at many different times and by different people before a file is printed. For instance, some file attributes can be assigned default values

according to usercode, some attributes can be given default values for a MARC or CANDE session, values can be assigned when printing a file, and values can be assigned when running a program. Because task and file attributes can be declared in so many ways, it is important to understand how conflicts between them are resolved.

The following list gives the order or precedence for file attributes in a task initiation from lowest priority to highest:

1. System default values
2. Attribute values specified by the PRINTDEFAULTS usercode attribute
3. CANDE *PDEF* command for a CANDE session
4. PRINTDEFAULTS task attribute
5. Attribute values specified in file declarations
6. Attribute values specified by file equation
7. Run-time assignments in program code

The following list gives the order of precedence for file attributes in a WFL *PRINT* statement from lowest priority to highest:

1. System default values
2. Attribute values specified in a PRINTDEFAULTS statement:
   a. Task attribute
   b. CANDE *PDEF* command for a CANDE session
   c. PRINTDEFAULTS usercode attribute in the USERDATAFILE
3. Attribute values maintained in the backup file control record
4. Attribute and print modifier values specified in the PRINT statement

Effect of a Halt/Load on the Print System

A halt/load operation has little effect on the Print System. All print requests are preserved intact across a halt/load. The states of all controlled devices, and their configurations, are unchanged. A halt/load affects only files that are in the process of printing.

If a halt/load occurs while a file is being printed, printing resumes from the previous checkpoint, if there is one. If no checkpoints were taken, printing resumes at the beginning of the file. (Note that this is one very good reason to configure your printing devices with checkpoints. Refer to "Checkpoints" in Section 8, "Establishing the Printing Environment.")

Sometimes special printing forms require realignment after a halt/load. If so, a message is sent to the ODT. The procedure for aligning forms is described in the *Print System Guide.*

Logging Print System Activity

Like all other activities on A Series systems, printing information is recorded in the system log file (SYSTEM/SUMLOG). Records are entered into the log file for the following printing activities:

- Creation and completion of print requests

  Each entry contains the following information:

  - Origin of the request (MCS number and LSN number, or ODT unit number)
  - Job or session number
  - Usercode
  - Chargecode
  - Print charge
  - Accesscode
  - Time of day
  - Print request number
  - Values of other print attributes, such as FORMID, NOTE, and AFTER

- Execution of printing commands (PS system commands and MARC commands)

  For every PS (Print System) command that is executed, a log entry contains the following:

  - Origin of the command (MCS number and station name, or ODT unit number)
  - Job or session number
  - Usercode
  - Time of day
  - Print request number
  - Text of the request

- Completion of file printing

  For every file that is printed, one log entry shows when printing began and another shows when it finished. If printing cannot be completed, the second log entry indicates the reason.

Refer to Section 11, "Logging System Events," for more information about the system log file and how to extract information from it.

Using the PRINTDEFAULTS Feature

Default values for printer file attributes can be established for a usercode and are stored in the USERDATAFILE. The file attribute default values specified for a usercode in the USERDATAFILE are inherited by every job or task run under that usercode. File attributes not specified in the USERDATAFILE use their system default values.

You assign default values for a usercode with the PRINTDEFAULTS usercode attribute. The default value for the PRINTDEFAULTS usercode attribute is null.

The default values specified in the USERDATAFILE are overridden by file attribute values specified

- By a CANDE *PDEF* command
- By the PRINTDEFAULTS task attribute
- In a file declaration
- By file equation
- At run time

Changes to the PRINTDEFAULTS usercode attribute in the USERDATAFILE do not affect active MARC and CANDE sessions; new values take effect when a new session begins. Also, if the PRINTDEFAULTS usercode attribute specifies a value for the DESTINATION file attribute, that value overrides a CANDEDESTNAME specification.

Establishing customized default values by usercode is often useful for file attributes such as DESTINATION. For instance, if a user requires a particular printer for the majority of his or her print jobs, you might want to assign that printer as the default for the DESTINATION file attribute for that user.

You can specify PRINTDEFAULTS values for a MARC session by using the PRDEF selection on the MARC *PS* menu. These values override any conflicting PRINTDEFAULTS values established for a usercode in the USERDATAFILE, and are applied to jobs and tasks initiated directly from MARC.

You can specify PRINTDEFAULTS values for your current CANDE session with the PDEF command. These values are inherited by every job and task initiated during the session. The values you specify with the PDEF command override any default values specified through PRINTDEFAULTS for your usercode in the USERDATAFILE.

The PRINTDEFAULTS value specified for a usercode in the USERDATAFILE is automatically inherited as the PRINTDEFAULTS value for a CANDE session when the session is initiated.

Transform Functions and Virtual Servers

Transform functions and virtual servers are both library functions. Each is used to alter, or somehow manipulate, records in a file between the time the file is created and the time it is printed.

Transform functions are general purpose; they can perform any type of processing you want. Virtual servers are more specific functions used to format records in a file so that they can be written on a nonstandard medium, such as microfiche. Virtual servers are, in effect, software devices. Device transforms cannot be associated with a virtual server.

Transform Functions

A transform function is a general-purpose function that can manipulate a print line before it is printed. Each print line in the printer file is the function's input; the function's output is sent to a printing device.

There are two types of transform function: device and file.

A device transform is associated with a specific device. Every file printed by that device has the device transform applied to it.

A file transform is a transform function that is applied to each line in a backup file before it is sent to a printer. A file transform is applied to a file regardless of the device that prints it.

If a file transform is specified for a backup file that is routed to a printer with a device transform, the file transform is applied before the device transform.

The system implements a transform function as a library entry point with standard parameters and defined results. The type of library depends on the function. For example, a function that inserts every print line into a database should be in a SHAREDBYALL library, but a function that reformats the layout of one print file for one user can be in a private library.

Transform functions usually fall into two general categories:

- Device-dependent functions, such as the insertion of sequences of escape and control characters into records, usually performed by device transforms
- Data-dependent functions, such as the decryption of records, usually performed by file transforms

Typically, transform functions are written to perform actions such as the following:

- Convert lowercase characters to uppercase, and vice versa.
- Reformat the data in a backup file.
- Save printed lines in a database.
- Copy printed lines to a security device.
- Insert control characters into output records.

Transform functions can also be categorized into standard (supplied by Unisys) and nonstandard (user-written) functions. Both types must adhere to a standard printer interface defined in the *Print System Guide.*

Standard Transform Functions

The Print System supplies several transform functions in the PRINTSUPPORT library. These are described as follows:

Transform Function	Use
AP9208	This device transform allows you to take advantage of the remote status reporting capability of the AP9208 printer. This function (or the AP9208 transform in the DEVICESUPPORT library) is required for files routed to a remote AP9208 printer.
B9246	This device transform alters data to meet the requirements of the B9246 printer. This function is required for files routed to a remote B9246 printer.
CANDEWRITER	This file transform formats data files into a format similar to that produced by the Command and Edit (CANDE) WRITE command. The actual printed format depends on the FILEKIND of the file. This function is used by default for all nonbackup files printed by the Work Flow Language (WFL) PRINT statement.
GENERALDC	This device transform formats data for printing on a remote (data comm) device. This function is used by default for all output directed to remote devices that are not configured with a device transform.
LOWERCASE	This file transform translates all uppercase characters in the input file to lowercase characters.
UPPERCASE	This file transform translates all lowercase characters in the input file to uppercase characters.

Note: The AP9208 transform in the PRINTSUPPORT library is provided for compatibility with software releases prior to Mark 3.9, and should not be confused with the AP9208 transform in the DEVICESUPPORT library that enables you to specify page formatting with the PAGECOMP file attribute.

DEVICESUPPORT Transform Functions

The Print System supplies several additional transform functions for remote printers in the DEVICESUPPORT library. These device transforms provide support for the Hewlett-Packard Printer Command Language (PCL) and the PostScript® page description languages, and allow you to control certain printer features using the PAGECOMP file attribute.

These device transforms also allow you to specify the connection used to transport the data, as explained later in this section.

PostScript is a registered trademark of Adobe Systems Inc.

The AP9208 and AP9215 device transforms provide the additional capability of remote status reporting. For more information about the Print System, refer to the *Print System Guide.*

Transform Function	Use
AP9208	Supports the Diablo® 630 internal emulation mode of the AP9208 printer. This function (or the AP9208 transform in the PRINTSUPPORT library) is required for files routed to an AP9208 printer.
AP9215	Supports the Diablo 630 internal emulation mode of the AP9215 printer.
AP9215PCL	Supports the PCL (Hewlett-Packard LaserJet® Plus) mode of the AP9215 printer.
AP9415PCL	Supports the PCL (Hewlett-Packard LaserJet Plus) mode of the AP9415 printer.
AP9415PS	Supports the PostScript mode of the AP9415 printer.
GENERICPCL	Supports any printer that recognizes level 4 of the Hewlett-Packard Printer Command Language (PCL). (For example, an AP9210 or a Hewlett-Packard LaserJet II.)
GENERICPS	Supports any printer that recognizes the PostScript page description language. (For example, an AP9210 with the PostScript option.)
GENERICTTY	Supports any printer that recognizes CR (carriage return), LF (line feed), and FF (form feed) codes.
MAFX	Supports certain printers (such as an AP1327, AP1329, AP9215-1, or Model 37) that recognize the EPSON® FX80 page description language when connected to a Micro A system using the XDLP connection type.

User-Written Transform Functions

A user-written function can be located in any library. For all data manipulation not handled by the standard transform functions, you must write your own functions.

Some transform functions are supplied with the Print System software, in the PRINTSUPPORT or DEVICESUPPORT libraries. For further information about these transforms, see the *Print System Guide.*

A user-written transform function can be located in any library. All transform functions must adhere to the standard printer interface described in the *Print System Guide.*

Diablo is a registered trademark of Xerox Corporation.
LaserJet is a registered trademark of Hewlett-Packard Company.
EPSON is a registered trademark of Seiko Epson Corporation.

The following situations are intended as examples of data manipulation that you can perform; they are by no means the only types of manipulations you can perform by writing your own transform functions:

- Printing or storing backup files created for standard devices on nonstandard output devices. This usually requires some data transformation, such as filtering control characters embedded in data.
- Processing a file to emulate special functions, such as SYSTEM/BACKUP utility functions.

Virtual Servers and Devices

Many sites have output requirements on devices that are not recognized by the master control program (MCP), such as a microfiche recorder. Such devices are referred to as offline devices.

To access offline devices, you must create a support library that contains procedures and functions to communicate with the devices. Each such library object (procedure or function) is called a virtual server. A virtual server routes backup files to printing or storage devices that are not supported by Unisys. The virtual server is a user-written library function that must be of type REAL. This function is passed information about various attributes of the backup file and the task that generated it.

A virtual server is not a statically defined entity; its contents can be any set of instructions. These instructions could be the requirements of a particular offline device, or they could be the requirements for data formatting at your site.

In most cases, part of every virtual server function contains a device driver; other parts of the function can be preprocessing instructions or any other processing you want performed.

The device that you configure with a virtual server is referred to as a *virtual device.* All print requests routed to the virtual device have their files processed by the virtual server. You must identify the virtual device to the Print System with the SERVER option of the PS CONFIGURE command. Once the device is known to the Print System, you can route output to it by naming it as the destination, just as you would any other output device.

Several other options of the PS CONFIGURE command can be applied to offline devices once they have been established. Offline devices can be configured as control stations, and can have limits and limit times. However, offline devices cannot be configured with the following options of the PS CONFIGURE command:

- CHECKPOINT
- FORMID
- LINESPERPAGE
- TRANSFORM

Remote Printers and BNA ReprintS

To find out whether BNA ReprintS is enabled or disabled on your system, use the PS BNA system command.

Some printers are capable of informing the host of their status. The ReprintS software is capable of accepting input from this type of printer. To make the format of the input expected by the remote server as device independent as possible, a generic format for input messages is available. Any input messages from a printer can then be modified into this generic format by means of a COMS processing item. The remote server can use the information in the generic format to obtain the status of the printer and display it on output screens when available.

A program is available to translate messages from an AP 9208 laser printer into the generic format. If you need status reporting for other printers, either you or a programmer must write a program to translate incoming printer messages into the expected input format. For more information on this type of program, refer to the *COMS Programming Guide.*

Operational Considerations

Operations tasks that involve the Print System consist mainly of monitoring the progress and status of print requests, and controlling peripheral devices. The following commands are most often used to perform these tasks:

```
PS DELETE
PS FORCE
PS MODIFY
PS NOTOK
PS OK
PS REQUEUE
PS SHOWREQUESTS
PS SKIP
PS STATUS
PS STOP
```

Specifying Where Backup Files Are Written

By default, all backup files are routed to the DISK family. Backup files can be routed to other disk families by using the SB (Substitute Backup) and DL (Disk Location) system commands.

Placing backup files on a family other than DISK frees up the DISK family and allows you to maintain backup files apart from other types of system-generated files. The SB command is used to prevent backup files from being written to the DISK and PACK families because these are used for other purposes.

Through the use of the BACKUPKIND file attribute, users can specify the type of output device used for writing the backup files generated by the jobs and tasks they initiate. With this attribute, users can specify DISK, PACK, TAPE, TAPE7, TAPE9,

TAPEPE, or DONTCARE. The default value is DONTCARE. The SB command can override all other BACKUPKIND values.

For example, assume the following SB command is in effect on your system:

```
SB TAPE = DISK
```

If a user specifies TAPE for the BACKUPKIND attribute of a backup file, that file is written to the DISK family instead of to tape.

If the value of the BACKUPKIND file attribute is DONTCARE, backup files are written to the DISK family, or to the device or family specified by the SB command or the SB and DL commands.

Backup File Structure

Line printer and card punch backup files are structured differently from image printer backup files. Line printer and card punch backup files are line-oriented; image printer backup files are page-oriented, and are called Virtual Static Imaging Device (VSID) backup files.

Naming Conventions for Backup Files

The system assigns file names to backup files on disk according the following default naming conventions:

- For printer backup files, the standard naming format is as follows:

  ```
  BD/<job number>/<task number>/<modified file name>
  ```

- For punch backup files, the standard naming format is as follows:

  ```
  BP/<job number>/<task number>/<modified file name>
  ```

BD (backup disk) and BP (backup punch) are the standard prefixes. The job number and task number in the standard backup file name are 7-digit fields, even though job and task numbers are currently limited to 4 digits (this allows for future expansion). The modified file name consists of 3 digits, ranging from 000 to 999, prefixed to the internal file name of the program output file or files. The 3-digit number is incremented each time a task or subtask opens a backup file, and is intended to prevent duplicate file names.

The general description of a standard backup file title does not show that tasks can generate subtasks, sub-subtasks, and so on, which can add additional nodes to the standard backup file name between the task number and the modified file name. Subtask numbers are also 7-digit fields of the same form and format as that of job numbers and task numbers.

Overriding Standard Names

You can override the standard backup file naming convention by using either the BDNAME task attribute to change the default prefix, or by using the USERBACKUPNAME file attribute and either the FILENAME or TITLE file attribute to rename the backup file entirely.

For example, the BDNAME task attribute is commonly used to make file identification easier; that is, you might want to replace the BD prefix with a prefix such as the name of the program, a usercode, or a department or section name.

The most common use of the USERBACKUPNAME and FILENAME or TITLE attributes is to rename a backup file when you save it.

Naming Tape Files

Backup files created on magnetic tape are labeled as follows:

```
BACKUP/<file name>
```

The file name is the name of the file as declared in the program that created it.

Section 8
Establishing the Printing Environment

Every site has different requirements for the output of information. Making that information easily accessible to the people who need it is probably the greatest concern when deciding how to configure your system for output. With regard to establishing your printing environment, you will need to consider the requirements of the system as a whole, then decide what configuration of individual devices meet those requirements.

You should consider the following aspects regarding the Print System:

- Options for configuration of printing devices
- Methods of processing print jobs
- Individual control of print jobs

This section lists tasks related to the administration of the Print System. For detailed information, refer to the *Print System Guide.*

Although in general the configuration of the printing environment is an administration activity, whereas the monitoring and controlling of printing devices and print jobs is an operations activity, the distinctions between administration and operations activities can vary from site to site. For example, at a small site the same person might perform both administration and operations tasks. At a large site, the system administrator might design the configuration of the printing environment, but delegate the actual implementation of that design to an experienced operator.

Generally, system administrators are responsible for the following tasks:

- Initializing the Print System
- Setting up the spooling process
- Setting the maximum number of print servers
- Specifying where backup files are written
- Establishing criteria for print request selection
- Specifying the following defaults for the Print System:
  - Whether the SYSTEM/BACKUPFILELIST file is compressed at each halt/load
  - Whether the Print System automatically creates and configures unknown remote devices
  - The default destination group for printing requests without a specific destination
  - The criteria used for printing job summary files
  - The default PAGECOMP declaration used to format job summary files
  - The default PAGECOMP declaration used to format backup files

  - The point when a print request is generated for a file
  - The printer type to be used for printing backup files
  - The removal of backup files for which the LOCKEDFILE file attribute has the value TRUE
  - The largest identification number that can be assigned to a print request
  - The ordering of print requests shown by a PS SHOWREQUESTS command
- Configuring output devices by doing the following:
  - Making them known to the Print System
  - Specifying PRINTERKIND values to indicate the supported page description languages
  - Specifying control users or stations
  - Setting volume limits and limit times
  - Configuring printers for special forms
  - Setting the page size (number of lines each form includes)
  - Specifying a default PAGECOMP declaration to be used on a device
  - Specifying whether a printer should take checkpoints, and setting the checkpoint interval
  - Setting up virtual servers to offline output devices
  - Specifying a transform function to be used on a printing device
  - Specifying whether a printer should suppress header and trailer pages by default
- Establishing the default destination printer group and other device groups
- Associating printing devices with specific users, specific input devices, or both
- Specifying the content and format of banner, header, and trailer pages
- Setting PRINTDEFAULTS by usercode
- Creating a load file of Print System commands

For detailed information on using the Print System, refer to the *Print System Guide* and the *Printing Utilities Guide.*

Most sites will have some sort of printer as a local (or site) device, though it is possible to use nothing but data comm (remote) devices. If there is no printer in the default pool, the job waits in the print queue.

The default printer pool is a group of devices used to service print requests that have not been assigned a specific destination. The default pool can include line printers, image printers, remote printers, virtual devices, and card punches. Each type of device can be either local or remote. Devices on another system (that is, accessed through BNA ReprintS) cannot belong to the default printer pool.

The default pool is automatically created when a device is assigned to it. The PS DEVICES command adds devices to and removes devices from the default pool. You can add or remove a single device or a device group. If a device group is specified, all the devices belonging to the group are added or removed.

Devices can be added to and removed from the default pool only with the PS DEVICES command. Adding a device to a group that belongs to the default pool does not automatically add the device to the default pool.

Print System Default Settings

In addition to assigning print requests to the default printer pool, you can also specify system default values for

- The default printer type for backup files
- The default printer type for individual job summaries
- The default print disposition (the time during the execution of a job at which backup files are queued)

The system uses these default values when a print request has no explicit assignment made through the use of file attributes. Use the PS DEFAULT command to display and assign default values for these three situations. Defaults set at the system level can be overridden by specific assignments to the related file attribute.

Default Printer Type

The default printer type is used to print backup files that have not been assigned a value for the PRINTERKIND or DESTINATION file attribute.

The DESTINATION file attribute does not have a default value; however, the PRINTERKIND file attribute defaults to the value DONTCARE, meaning that backup files and job summary files will print on the first available line or image printer.

With the PS DEFAULT command, you can set the default printer type to be a line printer or an image printer. You can also set it to a value of DONTCARE, but this is redundant if the PRINTERKIND file attribute is not used. The PS DEFAULT command setting also affects job summary files. Normally, job summary files are printed on the same device type that is specified for backup files.

The value of the PS DEFAULT command is preserved across halt/loads.

Default Print Disposition

The point at which a print request is generated for one or more output files from an executing process is known as the *print disposition*. The default print disposition is used if a value has not been assigned to the PRINTDISPOSITION file attribute for any backup file, and can be any of the following values:

Value	Meaning
DONTPRINT	Print request is not generated automatically.
CLOSE	Print request is generated when the file is closed.
EOT	Print request is generated on termination of the task or tasks that created the file.
EOJ	Print request is generated on termination of the job that created the file.

The PRINTDISPOSITION option of the PS DEFAULT command and the PRINTDISPOSITION file attribute both have EOJ (end-of-job) as their default value. If you do not set a default value through the PS DEFAULT command, then print requests are generated at the end of a job if another value for the attribute has not been assigned. The default PRINTDISPOSITION value is preserved across halt/loads.

If you use the DONTPRINT value, the Print System does not create print requests for the backup files. Backup files are created using the default naming conventions. With the DONTPRINT value, you can print selected files using the PRINT statement and remove the rest of the files manually.

Specifying How Print Requests Are Selected

All printing requests are assigned to a device according to the characteristics of the device and the print request. This device-driven selection process cannot be altered. Beyond this, however, you can specify how print requests are selected from the queue with the MCP option BACKUPBYJOBNR or the PS SELECTION command.

If you set the BACKUPBYJOBNR option (option 17) on your system, then backup files are selected out of the queue in order of ascending job number. If this option is set, it overrides any selection criteria available through the PS SELECTION command.

By using the PS SELECTION command, the order in which print requests are selected for printing can be based on the following criteria:

Value	Meaning
VOLUME	Selection is determined by print request size (in lines), with smaller requests printed first.
PRINTPRIORITY	Selection is determined by the value of this print modifier (range 0 through 99), with higher values chosen first.
REQUESTNUMBER	Selection is determined by the request number, with lower numbers chosen first.

The default value for the PS SELECTION command is VOLUME; therefore, if the BACKUPBYJOBNR option is reset and you have not used the PS SELECTION command to specify how print requests are selected, then requests are selected by size.

Whenever a PS SELECTION command is entered, the previous selection criteria are replaced. Furthermore, the specification of new selection criteria causes all print requests not yet serviced to be reevaluated based on the new criteria.

You can specify a single selection criterion or a hierarchical list of criteria. If you specify a hierarchy, print request selection is determined by the order of the listed criteria.

Selection by Volume

The VOLUME criterion causes the system to select print requests based on their size (in lines). Smaller requests are selected before larger ones. If two requests are the same size, the request number determines which request is selected first. (Request number 1000 is selected before request number 1001, and so on.) The VOLUME criterion is the Print System default.

Selection by Priority

The PRINTPRIORITY criterion causes the system to select print requests based on the value of the PRINTPRIORITY print modifier. Accepted values are in the range 0 through 99, with higher values chosen first. The default value for this modifier is 50.

If two requests have the same PRINTPRIORITY value, the request number determines which request is selected first, with lower numbers selected before higher numbers.

If PRINTPRIORITY is the only selection criterion, or the first criterion in a hierarchical list, then print requests with a PRINTPRIORITY value of 0 (zero) are not serviced. A request with a PRINTPRIORITY value of 0 can be selected only if the PRINTPRIORITY print modifier is changed to a value other than 0 (zero) with the PS MODIFY command, or the print request is forced with the PS FORCE command.

Selection by Request Number

The REQUESTNUMBER criterion causes the system to select print requests based on the print request number. The Print System assigns request numbers in ascending order, and the lowest number in the print queue is selected first. If you want print requests to be selected on a first-come, first-served basis, then use this selection criterion.

Hierarchical Selection

You can specify more than one selection criterion. However, because the request numbers of two print requests cannot be equal, there are only two hierarchies to consider:

- Print priority first, followed by volume
- Volume first, followed by print priority

When you specify a selection criteria hierarchy, the criterion listed first has precedence. If two or more requests have the same value for the first criterion, then the second criterion is used. If two or more requests have the same value for the second criterion, the request with the lowest request number is selected first.

Printer Groups

Printing devices can be organized into logical groups that can be referred to by a single name. A device group can be composed of any number of virtual devices, local (site) devices, and remote (data comm) devices, but is limited to one device linked to another system through BNA. A device can belong to any number of device groups.

A local device can be a line printer, image printer, or card punch. A remote device can be a line printer only. The device linked through BNA can be either local or remote, and can be a line printer, image printer, or card punch.

When a group name is specified as the destination for a print request, a device is chosen in the following order:

1. Any available virtual device
2. Any available site (local) printer in the group
3. Any available remote printer in the group
4. The device linked through BNA, if one belongs to the group

If neither a local printer nor a remote printer is available, the print request is transferred to the BNA host. Note that the request is transferred to the other host even though it might not be printable immediately at the other host.

Device groups are created, modified, and deleted through the PS GROUP command. You can redefine any established group with another PS GROUP command. The new command simply overwrites the old device assignments. Adding and removing individual devices does not affect other members of the group.

Device and User Associations

Input devices and usercodes can be associated with specific output devices or device groups. An association causes all print requests originating from a particular input device or usercode to be serviced by a particular printing device or group of devices by default. Essentially, the associated output devices comprise a customized default printer

pool for an input device or a usercode. Any number of input devices and usercodes can be associated with an output device or device group.

A particular output device or device group can be configured with both input device and usercode associations and there is no restriction on how many of each. However, an input device association takes precedence over a usercode association if there is a conflict. The DESTINATION file attribute overrides all device and usercode associations.

Device and usercode associations are created with the PS ASSOCIATE command. Input device and user associations known to the system, and the output devices that make up each association, are displayed by the PS ASSOCIATE command. By including a device name or group name in the command, you can display all the input device and user associations for a particular output device or device group.

Input Device Associations

An input device can be a data comm station, ODT, or card reader. If you use the *WITH <input device list>* form of the PS ASSOCIATE command, the associated output device or device group can also service requests that originate from unassociated input devices. The *ONLY WITH* form of the command restricts the output device or device group to servicing only requests originating from associated input devices.

Usercode Associations

If you use the *WITH USER* form of the PS ASSOCIATE command, the associated output device or device group can also service requests that originate from unassociated usercodes. The *ONLY WITH USER* form of the command restricts the output device or device group to servicing only requests originating from associated usercodes.

Input device and usercode associations can be added to or removed from a device or device group without affecting previously specified associations for the output device or device group.

To associate an output device with both input devices and usercodes, first you must create an input device association and then add the usercode association, or create the usercode association and then add the input device association.

Starting and Stopping the Print System

The PS QUIT and PS RESTART commands enable you to update the PrintS software or change versions of the Print System or PRINTSUPPORT library without disrupting the rest of the system. Note that the MCP is not designed to run without PrintS.

After you enter a PS QUIT command, no new requests for use of the Print System, its utilities, or the PRINTSUPPORT library are accepted. All remote server, print server, and BNA ReprintS tasks terminate immediately. If you are running MARC, the PS QUIT command causes MARC to delink from the PRINTSUPPORT library.

Backup Processor sessions, if any, are allowed to complete normally. If any Backup Processor sessions are running, the PRINTSUPPORT library becomes an active task and then waits until all the Backup Processor sessions are complete.

The PS RESTART command is used to restore the Print System functions after a PS QUIT command. PS RESTART can be executed even when Backup Processor sessions are running with the old invocation of the PRINTSUPPORT library.

While the Print System is stopped, programs requiring access to the PRINTSUPPORT library (mainly those that access backup files) display the following message:

```
WAITING FOR PRINTSUPPORT TO INITIALIZE
```

Execution of these programs is resumed once the Print System has been restarted (using the PS RESTART command).

While the Print System is stopped, the SYSTEM/BACKUPFILELIST file should not be changed or removed. If this file is not preserved, all printer backup files that were open when the Print System was stopped are lost.

Inquiring About Device Characteristics

The characteristics of all printing devices known to the system can be displayed with the PS DEVICES command. The display always lists the device type, the accessibility of the device, and the request limits for the device. The following additional items and their values are displayed if configured for a device:

- Form identifier
- Transform function
- Limit times
- Control users, stations, or ODTs
- Checkpoint
- Lines per page

The PS DEVICES command enables you to display the characteristics of a specific device or all devices known to the system. The PS GROUP command displays the names of device groups known to the system and the devices belonging to each group. You also can display information about a particular group by including the group name.

Configuring Individual Devices

The preceding discussion focused on the means of specifying a printing strategy for the system as a whole. Beyond that level, you can assign characteristics to individual printing devices that allow those devices to be used only in specific ways or for specific purposes.

Volume Controls

You can configure any printing device to accept only print requests that fall within a certain size range. You can also specify a time range during which the size restriction applies.

Configuring Size Limits Only

You can configure printers with a print request size limit (in lines) in different ways. You can

- Set the minimum size of a print request that can be accepted by a particular printer.
- Set the maximum size of a print request that can be accepted by a particular printer.
- Set a size range within which a print request must fall to be accepted by a particular printer.

Restricting Size Limits to Specific Times

You can restrict a size limit for a device or device group to a specific time period. If a device or device group is not configured with a size limit, the Print System ignores a limit-time specification.

A time range is given using a 24-hour clock. A time range is any interval of fewer than 24 hours, but the range can extend across midnight.

Printer Page Sizes

Certain circumstances, such as printing on special forms, make it desirable to configure a printer with a page size. If a printer is configured with a page size, the PS SHOWREQUESTS command displays print request size in pages rather than lines.

To configure a printer device or group with a particular page size, use the following command:

```
PS CONFIG <device specifier> LINESPERPAGE = <integer>
```

The device specifier is the name of the printer or group. You can abbreviate LINESPERPAGE as LINES.

Note: *The LINESPERPAGE option does not perform the same function as that of the PAGESIZE file attribute. For a description of the PAGESIZE file attribute, refer to the* File Attributes Reference Manual.

Checkpoints

The use of checkpoints on a printer prevents unnecessary repetition of print requests or portions of print requests if a printer is stopped for some reason. For instance, if a

halt/load occurs, all files that were printing are interrupted and have to be restarted when the system is brought up again. If checkpoints are used, files are restarted from the last checkpoint; otherwise, files are restarted from the beginning of the file.

Devices can be configured so that checkpoints are mandatory or optional, and with specific checkpoint intervals (the frequency of checkpoints). The interval can be defined either in lines or in pages; checkpointing by lines is the default.

You can assign checkpoints using the CHECKPOINT option of the PS CONFIGURE command. If the CHECKPOINT option is unassigned or has been removed for a device, the default checkpoint interval of 10,000 lines is used for print requests whose CHECKPOINT file attribute is TRUE. Print requests smaller than 10,000 lines are restarted from the beginning of the file. If the CHECKPOINT file attribute is FALSE, checkpointing is not used unless a printer is configured for mandatory checkpoints.

An optional checkpoint configuration means that a printer uses checkpoints only if the CHECKPOINT file attribute for a backup file is TRUE.

You can specify a constant checkpoint interval or an initial interval followed by a subsequent interval.

A mandatory checkpoint configuration means that a printer uses checkpoints no matter whether the CHECKPOINT file attribute for a backup file is TRUE or FALSE.

You can specify either a constant checkpoint interval, or you can specify an initial interval followed by a subsequent interval.

Configuring a Printer for Special Forms

If a print request requires a special form, such as a paycheck form or an invoice form, both the printer and the print request must have the same form identifier, or FORMID, before the request can be printed. A FORMID can be associated with a line printer, image printer, or card punch.

Once a device has been configured with a FORMID value, it can accept only files whose FORMID file attribute matches the value specified by the PS CONFIGURE FORMID command.

Transform Functions

You can configure a printer with a transform function, in which case the function is referred to as a device transform. (Transform functions also can be associated with files and are called file transforms.) Device transforms are applied to each line of every file serviced by the printer. For some printers, and for some applications, it is necessary to configure the printer with a transform function.

For more information about transform functions, including descriptions of the standard transform functions supplied with the Print System software, refer to Section 7, "Understanding the Print System."

Control Stations and Users

You can configure output devices to have a control station or a control user. If a device has a control station, PS commands affecting that device can be entered only from that station. If a device has a control user, PS commands affecting the device can be entered only by that usercode. The usercode of a control user can be nonprivileged.

When an output device has a control station, privileged users and MARC users with SYSTEMUSER status can affect the output device only if they enter commands from the control station.

When an output device has a control user, no other user can enter commands affecting the controlled output device. Specifying a control user is a convenient way to give device control to a user without having to give that user privileged status.

An output device can have multiple control stations or control users. Control over an output device or device group can be assigned to the following:

- Any valid usercode
- A MARC station
- An ODT

When device control is given to a usercode, commands affecting the output device can be entered from any input device where the user is logged on.

A *PS CONFIG CONTROL* command can be entered from any ODT at any time to change the control for a device if the current control station, ODT, or user becomes unavailable.

If an output device or device group is uncontrolled (that is, a control assignment was removed or control was never explicitly assigned), Print System commands affecting the output device or device group can be entered from any ODT, by a MARC *SYSTEMUSER,* or by a privileged user.

Formatting Header, Banner, and Trailer Pages

Header, banner, and trailer pages enable users to easily distinguish different print requests when they appear on the printer. A header page indicates the beginning of a print request, and a trailer page indicates the end. A banner page marks the beginning of each file (if there is more than one) in a print request.

Header pages are usually formatted so that they identify the originator of the request and the system that processed it, as well as indicating the job name and number, and the date and time the request was printed. Trailer pages usually contain the character string END and the job number. Banner pages most often contain a special note concerning the file.

You can display the current format description of the header, banner, or trailer page by entering the command PS HEADER, PS BANNER, or PS TRAILER respectively. The displays for all three commands have the same layout. If a page has no format

information specified, the system informs you. Refer to the *Print System Guide* for full details about how to define the page formats.

Default Page Formats

The system prints header, banner, and trailer pages for all print requests according to the default format if you do not specify your own formats. The default formats are described in the following text.

Header Page

The default header page contains the following content items, printed in the order in which they are listed:

1. Usercode, in block characters
2. Accesscode, in block characters
3. Chargecode, in block characters
4. Job or session number, in block characters
5. Job name, in block characters
6. Job name, in uppercase characters
7. Origin, in uppercase characters
8. System identification, in uppercase characters
9. Time and date the page was printed, in uppercase characters

Banner Page

The banner page is printed if the BANNER file attribute is TRUE. The default banner page format is determined as follows:

- If the NOTE file attribute value is a null string, the title of the backup file is printed in block characters.
- If a string is specified for the NOTE file attribute, that string is printed in block characters.

Trailer Page

The default trailer page format prints the STATUSLINE content item of the page definition at the top and bottom of the page in block characters.

Defining a Page

To define a header, banner, or trailer page you create a page specification using the PS HEADER, PS BANNER, or PS TRAILER command as appropriate. The page

specification consists of numbered lines each containing one of the following items: content items, format items, and spacing items. These items are described later in this section.

Every content item has an associated format item; the default format item is uppercase characters. You can add content items and spacing items to a page that is already defined. For this reason, it is a good idea to number items in increments of 10 or more.) Specifying a number already used in the page specification replaces that content or spacing item with the new item with the same number.

Content Items

If you want to create your own formats for header, banner, and trailer pages, their content is defined by the following content items:

ACCESSCODE	CHARGECODE	DEVICE
FILENAME	JOB NAME	JOB NUMBER
JOB QUEUE	NOTE	ORIGIN
PRINTCHARGE	REQUESTNOTE	REQUESTNUM
STATUSLINE	SYSTEMID	TIME
USERCODE	<string constant>	

You can specify any number of content items for a page (including the same item more than once) in any order you want. Refer to the *Print System Guide* for instructions on creating your own formats. The guide also contains examples of page format specifications.

Spacing Items

Spacing items separate content items from one another. You have the following two choices:

Spacing Item	Function
SPACE <n>	Specifies the number of blank lines to print
PAGE	Causes a skip to the top of the next page

Format Clauses

You can specify one of four format clauses for each content item:

Clause	Function
BLOCK	Prints the associated content item in block characters
UPPER	Prints the associated content item in uppercase characters
LOWER	Prints the associated content item in lowercase characters
CALL	Calls a user-defined format function

User-Defined Format Functions

You can write your own formatting functions for the header, banner, and trailer pages. Three procedures must be defined and exported in a user-written library. This library must be associated with the library function name PSFORMATSUPPORT through the SL (System Library) system command. The three procedure names are

- HEADER_FORMAT
- BANNER_FORMAT
- TRAILER_FORMAT

You must use these procedure names as the names of the formatting functions you create. Each of these procedures can describe any format the printing device is capable of producing. If you choose to define your own formatting, then you must define all three functions.

Each formatting function must accept the following parameters in the same order:

- A real value to accept the content item.
- A one-dimensional array to contain the value of the content item in string form. The first byte of the string contains the length of the string.
- A two-dimensional array through which formatted output is returned. Each row in the array contains the information for one print line.
- A real value indicating the number of print lines in the two-dimensional array.

As an example, assume that these three formatting functions have been created in a library file called SYMBOL/PSFORMATS. Next, you compile that file into a code file called SYSTEM/PSFORMATS. This code file is then used in the SL system command, for example:

```
SL PSFORMATSUPPORT = SYSTEM/PSFORMATS
```

Once your site-defined formatting functions have been placed in the PSFORMATSUPPORT library, those functions can be accessed by using the CALL format clause for any content item that you want processed by one of the formatting functions. Note that if any changes are made to these formatting functions, or if any header, banner, or trailer page content items that use those functions are changed, then the Print System must be reinitialized for those changes to take effect.

Section 9
Managing Processes at the Operating System Level

An A Series system provides process control at two levels:

- The operating system level.

  These controls involve the system's scheduling and suspending algorithms and are automatic. This level includes the memory management factors.

- The job and task level.

  This level involves the use of job queues to control WFL jobs and the use of MCS session control commands to affect tasks.

This section discusses control of the workload at the operating system level, and Section 10 provides information on controlling the workload at the level of job and task execution.

The operating system has certain controls it applies to all processes, regardless of the source of initiation. As soon as a job is submitted for execution, the operating system can influence what happens to the process through three factors:

- Priority
- Scheduling
- Suspension

This section considers the use of these three factors, as well as the use of resident programs and control programs.

If system resources such as memory or ASDs are in short supply, the first step the operating system takes to control contention for resources is to *schedule* processes. Scheduling a process means that the system prevents the process from entering the job mix until enough resources are available. If processes that are already active create more demand than can be handled, then the operating system will start *suspending* processes. Suspension means that a process that has already begun executing is temporarily halted. This feature allows the highest priority processes to continue until completion, whereupon suspended processes will be resumed.

Whether or not a process is scheduled or suspended is based on the priority assigned to the process. System priorities are established by several operating system routines and by the values assigned to a number of system options, memory factors, and task attributes. In some instances, the operating system uses these values to make decisions concerning which processes it will allow to have access to the processor or other system resources.

Scheduling of processes can be performed on two levels: automatically by the operating system scheduling algorithms, or manually through the use of system commands.

Process suspension usually occurs automatically, but can be influenced by the settings of certain memory management factors, and by the value of the PRIORITY task attribute for a process.

Process Priority

The operating system uses the concept of priority as a basis for allocating resources. In this way, the operating system determines which of the many active processes on the system should have access to the processor at any given time. When two or more processes are in contention for use of the processor, the process with the highest priority is chosen. This process can continue using the processor until a higher priority process needs the processor. A process can also give up the processor if it needs to wait for an I/O operation to complete.

The operating system also uses the priority of a process in the following ways:

- To regulate the ability of a process to overlay memory areas that are being used by other active processes. Controlling the use of memory is discussed under "PRIORITY Memory Factor" in this section.
- To help determine the order in which scheduled processes are initiated. This topic is discussed under "Process Scheduling" in this section.
- To help determine the order in which processes are suspended. This topic is discussed under "Process Suspension" in this section.
- To determine the following factors related to job queues:
  - Whether a job with a certain priority is allowed in a given job queue. For more information, refer to "QFACTMATCHING Algorithm" and "Job Queue Verification" in Section 10, "Managing the System Workload."
  - The order in which jobs are listed in a job queue. For information, refer to "Ordering Jobs in a Queue" in Section 10, "Managing the System Workload."
  - The order in which eligible jobs are selected from different job queues. For information, refer to "Selecting Jobs from Queues" in Section 10, "Managing the System Workload."

The priority of a process is viewed by the operating system in several ways. Different types of processes are automatically assigned to priority categories. The definition of categories, or the classification of a process, in general is not adjustable. You can, however, influence process priority at other levels. You can use the PRIORITY task attribute or the PR (Priority) system command.

Fixed Priority Categories

Each process on the system belongs to one of several fixed categories of processes, and each category has a different priority. These categories are described in the following list and are presented in order from highest to lowest priority:

1. Invisible independent runners and some special independent runners

   Most visible independent runners, such as LIBRARY/MAINTENANCE, do not belong to this category.

2. Control programs, message control systems (MCSs), and WFL jobs

   A program becomes a control program when you mark it as such using the CONTROL option of the MP (Mark Program) system command. WFL jobs are accorded a special priority because they are used mainly to initiate tasks and, in themselves, do not usually use much processor time.

   For more information about control programs, refer to "Control Programs" in this section.

3. Regular user processes

   This category includes tasks that are initiated by WFL jobs or by CANDE or MARC commands. In general, most user-initiated programs will belong to this category except for programs that previously have been marked as control programs.

Temporary Priority Categories

In addition to fixed priority categories, there are several priority categories to which a process can temporarily belong. For example, a process that is using a critical system resource, such as a lock, can be temporarily assigned to a high-priority category. For the most part, these temporary categories are not visible to the user.

One temporary category of interest is that of a discontinued process. Discontinued processes are put into a high-priority category so that they can be dealt with quickly. Processes that do not get discontinued quickly could be waiting either on a program dump or on an unexpected condition that was encountered in the operating system that discontinues a process. If the delay is caused by the process issuing a program dump, the dump can be discontinued with a second DS command.

If the second DS command does not take effect quickly, then you can remove the process from its temporary high-priority category with the PR (Priority) system command. Setting the priority of the process to zero causes the process to be removed from the priority category. The process remains in the mix until the next halt/load, but uses system resources at a very low rate because of its low priority.

Note that the PR command does not affect the priority category of any other type of process. Only processes that are being discontinued can be shifted to a lower priority category by this means.

Methods of Controlling Priority

Even though you generally want high-priority processes to have quicker access to system resources, situations can arise where the system runs slowly because one or more high-priority processes are dominating the processors.

You can confirm such a situation by using the A SORT CPU system command or by referring to the TI (Times) system command display. To obtain this display, enter the

command *<mix number> TI*, where <mix number> is the number that the operating system assigns to the process about which you are enquiring. The display entries labeled PROCESS and ELAPSED show the accumulated processor time and elapsed time respectively for one or more specified processes in the mix. If a process has very high processor time compared to its elapsed time, it might be dominating the processor.

Another part of the TI display that might be helpful is the entry labeled READYQ; this shows the amount of time that the process has spent waiting for service from a processor.

If the process is caught in a loop, it might be necessary to discontinue the process with a DS (Discontinue) system command or suspend it with a ST (Stop) system command to improve response time for other processes. For more information about the TI command, refer to the *System Commands Reference Manual.*

You have three means of controlling the selection of processes based on priority:

- Through the PRIORITY memory factor
- Through the PRIORITY task attribute (can be affected at both the usercode and the task level)
- Through the PR system command, which makes an assignment to the PRIORITY attribute

PRIORITY Memory Factor

The PRIORITY memory factor enables you to control the degree to which high-priority processes are preferred over lower-priority processes for access to main memory. The values for this attribute can be in the range 0 through 100. The default value is 100.

If the value of this memory factor is zero, then all processes have equal access to main memory.

Setting this factor to 100 allows a high-priority process to use as much memory as it needs while lower priority processes have access only to the memory the high-priority process is not using. This setting allows a high-priority process to execute without becoming memory bound. However, this value can be undesirable in some cases. If high-priority processes that use very large amounts of memory are run, they can prevent other processes from accessing enough memory to execute. This situation can be remedied by setting the PRIORITY memory factor to a lower value.

Values in between 0 and 100 signify the degree to which high priority processes are favored. Finding the optimum setting for your site is a matter of trial and observation. In general, the default value of 100 usually works quite well. Note that it is usually a good idea not to change the system defaults without a good reason.

PRIORITY Task Attribute

Within each of the fixed priority categories, the value of the PRIORITY task attribute determines the rank of any individual process. This task attribute can have a value

from 0 through 99, with a higher value signifying a higher priority. The PRIORITY task attribute is not used when comparing processes that belong to different categories; for example, a control program with a PRIORITY value of 1 has a higher priority than an ordinary user process with a PRIORITY value of 99.

When several processes in the same priority category are ready to use the processor, the process having the highest value for the PRIORITY task attribute is chosen. A lower-priority process cannot take the processor away from a higher-priority process, even if the difference in values is 1. The lower priority process must wait until the higher-priority process is finished using the processor.

The PRIORITY task attribute can get its value from several sources. A default priority value can be assigned to a usercode through the PRIORITY usercode attribute. This value is inherited by CANDE and MARC sessions at log-on time and is, in turn, inherited by any processes initiated by either MARC or CANDE commands.

An explicit assignment to the PRIORITY task attribute when a process is initiated overrides the value of the usercode attribute.

The PRIORITY task attribute of a WFL job can be affected by the job queue mechanism. A WFL job is not accepted into a particular job queue if it requests a priority higher than the queue allows. The job queue can also specify a default priority that is used for the jobs in that queue. The PRIORITY value of the WFL job is inherited by all its descendant tasks unless you specifically override the value by using task equation or task assignment statements. This inherited value establishes a maximum priority limit for descendant tasks—they can be assigned only a lower value.

You can change the value of the PRIORITY task attribute after the process is initiated by using the PR (Priority) system command. The change in priority is effective immediately.

PR (Priority) System Command

The PR system command is used primarily to change the value of the task's PRIORITY attribute, and thereby alter the priority of processes that are in the mix, in the queue, or in the schedule.

You can use this command to speed up an urgent process, to slow down a process that is dominating the processor, or to remove a discontinued process from its special priority category.

Fine Priority

The fine priority of a process is used to choose among processes that belong to the same priority category and have the same value for the PRIORITY task attribute. The operating system computes the fine priority of a process based on the following considerations:

- Processes that have not accumulated much processor time are favored. This gives faster service to shorter processes.
- Processes whose processor usage is burst oriented are favored. The operating system determines which processes are burst oriented by evaluating their behavior. This strategy improves response time for interactive programs.
- Processes that have been waiting in the ready queue for a long time are favored. For information about the ready queue, refer to the *Task Management Guide*.

Process Scheduling

The operating system evaluates each new process entering the mix to determine whether it should be initiated immediately or scheduled. There are a number of reasons why a process can be scheduled, such as insufficient memory, insufficient ASDs, or an excessive overlay rate. Some processes cannot be scheduled under particular circumstances.

A process that has been scheduled appears in the Scheduled Entries list. This list is displayed by the S (Scheduled Entries) system command.

Immunity to Scheduling

No process is totally immune to scheduling–an insufficient amount of ASDs causes any process to be scheduled. In all other situations, operating system invisible independent runners are not scheduled, regardless of their memory size estimates. Additionally, any code file that has been designated as a control program cannot be scheduled for a reason other than insufficient ASDs. Refer to "Control Programs" later in this section for more information.

An MCS (for instance, COMS or CANDE) does not have any special immunity to being scheduled because it does not become an MCS until after it is initiated.

Scheduling Caused by Insufficient ASDs

When a new process is initiated, the operating system checks the amount of available ASDs. If the amount of available ASDs is less than 5 percent of the total, then all new processes are scheduled. No type of process is immune to being scheduled because of insufficient ASDs.

As active processes terminate, ASDs are released and become available. When the amount of available ASDs exceeds 5 percent, scheduled processes again become eligible

for initiation. The rules for determining which process is initiated first are discussed under "Resuming Scheduled Processes" later in this section.

If the system frequently runs out of ASDs, then you should consider increasing the size of the ASD table. This table resides in main memory; therefore, increasing the ASD table size reduces the amount of memory available to processes. Memory management is most efficient when the ASD table is just large enough to provide sufficient ASDs for all the processes on the system.

Use the ASD (actual segment descriptor) system command to display information about the ASD table or to change the size of the ASD table. If a process has been scheduled, look for a system message displaying the reason for the scheduling. The scheduling reason might also appear in a message when the operator performs the S (Scheduled Entries) system command.

If you are trying to determine why a process has been scheduled, refer to the entry in the display that cites the available percentage of the total number of ASDs. If the value displayed is less than 5 percent, then this is the reason. Changes to the ASD table size do not take effect until the next halt/load.

For more information about the ASD command and the information it displays, refer to the *System Commands Reference Manual.*

Scheduling Caused by Insufficient Memory

The operating system can schedule a process if the system estimates that memory resources are too heavily used to allow a new process to be executed efficiently. The operating system compares the memory estimate for each new process with the currently available memory. The process is scheduled if enough memory is not available.

You can affect the likelihood of a process being scheduled. This capability is provided through the following parameters of the SF (Set Factors) system command:

- OLAYGOAL
- AVAILMIN
- FACTOR

The operating system multiplies the available memory estimate by the value of the FACTOR parameter before deciding whether to schedule a process. Setting FACTOR to a high number makes it more likely that any given process will be initiated immediately rather than scheduled.

If you set the OLAYGOAL memory management parameter to a nonzero value, the system initiates a memory management procedure called the second chance overlay mechanism (SCOM). When SCOM is running, you can use the AVAILMIN memory management parameter to specify a percentage of memory that should be kept available at all times.

The basic purpose of setting AVAILMIN is to prevent "thrashing," a situation in which the system spends more time in overlaying memory than it does in processing data.

When a new process is submitted, the system considers the memory estimate for the process and the amount of memory that is currently available. If initiating the process would cause the available memory to drop below the minimum specified by AVAILMIN, then the system schedules the process, temporarily preventing its initiation. The higher you set the AVAILMIN value, the more likely the system is to schedule processes.

A lower AVAILMIN value means that thrashing is more likely because there might be too many processes competing for a limited amount of available memory space.

If necessary, the system also suspends (temporarily halts) processes that have already been initiated. The system does this to maintain the amount of available memory required by the AVAILMIN parameter. If any process is suspended for this reason, the system schedules all new processes that are submitted. This type of suspension is discussed further under "Suspending for Memory Control" in this section.

For a full description of the SF command, refer to the *System Commands Reference Manual.*

Scheduling Caused by Insufficient System Log Space

By default, the system log resides on the halt/load family, but you can place the log on any existing, online family by using the DL LOG command. If the system log family runs out of space, the LOGHANDLER visible independent runner is suspended and appears in the W (Waiting Mix Entries) system command display. While LOGHANDLER is suspended, all new processes are scheduled.

The only way to resume the LOGHANDLER runner is to free space on the system log family. Most of the time, this situation arises because the system log is full, or because you are keeping too many old system logs on the same family with the current log file. To remedy the situation, either use a new family as the system log family, or transfer old log files or other nonessential files to another family or to tape.

Note that a log transfer operation can further aggravate this problem.

Scheduling Caused by Insufficient Overlay Space

The OLAYSCOUT visible independent runner is responsible for performing overlays. By default, the overlay is performed to the halt/load family, but you can use the DL OVERLAY command to place the overlay files on any existing online disk families. You can also have multiple overlay families.

If the overlay families run short of disk space, OLAYSCOUT might be unable to acquire overlay space for new processes. The operating system therefore schedules new processes until the situation is remedied.

To remedy the situation, you can

- Add another disk to an overlay family.
- Designate another or additional overlay families.

- Remove files from the overlay families.

Scheduling Caused by an Excessive Overlay Rate

When SCOM is not in effect (OLAYGOAL = 0), the system calculates a maximum overlay rate based on the amount of overlays the I/O hardware can handle without becoming overloaded. If the current overlay rate for the system rises above this maximum overlay rate, the operating system schedules any new processes that are submitted. Scheduled processes become candidates for initiation when the overlay rate drops below the maximum.

Scheduling caused by an excessive overlay rate can be avoided only by using the second chance overlay mechanism. To determine whether processes are being scheduled because of an excessive overlay rate, use the U (Utilization) system command. If the resulting display shows a high percentage of processor time for the OTHER PBIT and SEARCH entries, the overlay rate is probably excessive.

Scheduling caused by excessive overlay rate causes a system message to be displayed. The scheduling reason might also appear in a message when the operator performs the S (Scheduled Entries) system command.

Scheduling Initiated by System Commands

The scheduling of processes can be manipulated manually with particular system commands (as compared to automatic scheduling due to memory considerations). Processes can be scheduled by the following system commands:

- HS (Hold Schedule)
- CM (Change MCP)
- RECONFIGURE (Reconfigure System)

HS (Hold Schedule) System Command

The HS system command prevents the initiation of all processes for as long as the command is in effect. The FS (Force Schedule) system command can override the schedule hold for specific processes.

You might want to enter the HS command for one of the following reasons:

- Immediately after a halt/load, to prevent any processes from being initiated until you complete various actions. For instance, you might want to power up a particular disk unit that you expect many processes to use.
- To determine the job that is causing the system to crash. It occasionally happens that a process triggers some unexpected condition in the system that only a halt/load can cure. If a WFL job initiated the process, then the WFL job automatically restarts after the halt/load, thus crashing the system again. This loop could continue indefinitely. Resetting the AUTORECOVERY MCP option prevents this.

In this situation, you can enter the HS command immediately after the halt/load to prevent the process from being initiated. Then you can use the FS command to force initiation of each scheduled process, one by one, until the guilty process is found. To terminate the process, you can use a DS (Discontinue) system command.

You might prefer to use the primitive ??HS form of the HS command, because the primitive form takes effect even if the system is not responding to nonprimitive system commands.

CM (Change MCP) System Command

The CM command can cause a different copy of the MCP to be executed instead of the one currently running. The system automatically halt/loads when a change to a new MCP code file is made.

The change to the new MCP takes place when a null mix is achieved (that is, when no jobs or tasks are active). If a null mix has not been generated before the CM command is entered, then a process called CHANGEMCP appears in the waiting entries and any new processes are scheduled.

The FS (Force Schedule) system command can override the schedule hold for this process.

If you want the MCP change to take place immediately, then you can use the ??CM primitive system command. This command causes a halt/load and changes the MCP immediately, without waiting for a null mix.

RECONFIGURE (Reconfigure System) System Command

You can use the RECONFIGURE command to regroup the hardware resources of the system according to the specifications contained in the system configuration file. If the *NOW* option is not included in the RECONFIGURE command, then the reconfiguration does not occur until all groups affected by the reconfiguration have a null mix. While the system is waiting for a null mix, any new processes are scheduled.

Any processes that are running continue to execute until they reach an end-of-task (EOT) or end-of-job (EOJ) condition. New tasks are not started. The system reconfiguration then takes place.

The FS (Force Schedule) system command can override the schedule hold for this process.

Resuming Scheduled Processes

When several processes have been scheduled, the operating system sorts them into a list in order of importance. Only the process at the head of the list is a candidate for immediate initiation. This process is initiated when sufficient memory becomes available. The next process in the list then becomes eligible for initiation.

The operating system uses several factors to sort the list of scheduled processes. These factors are presented from most to least important in the following list. Note that independent runners are rarely, if ever, scheduled.

1. WFL jobs are chosen before any other scheduled processes. This is done because WFL jobs exist primarily to initiate other tasks. Entering the WFL job into the mix allows these tasks to be evaluated individually for scheduling.
2. Processes with higher PRIORITY task attribute values are chosen before processes with lower PRIORITY values.
3. Processes that have been waiting for a long time are chosen before processes that have been waiting a shorter time.

The factors are considered in order of importance. If the system has to choose between two WFL jobs, for example, the value of the PRIORITY task attribute for each job is the next deciding factor. If the PRIORITY values are identical, the system selects the job that has been waiting longer. However, if the system has to choose between a WFL job and some other type of job, the WFL job is always selected first.

The ordering of the schedule list can be overridden by the FS (Force Schedule) system command. This command forces the operating system to initiate the process immediately. If necessary, the operating system suspends existing processes to make room for the new process.

Process Suspension

The operating system suspends an active process if the demand for memory or ASDs is greater than what is currently available, or if overlay space is not available.

Suspending for Memory Control

The system can suspend processes automatically if a shortage of available memory occurs. Suspending processes increases the amount of available memory so that the remaining processes in the mix can execute efficiently.

A process can be suspended because of either insufficient memory or insufficient ASDs.

Processes can be suspended for many reasons. For more information about these types of situations, refer to the *Task Management Guide*

A suspended process appears in the Waiting Entries display with the following message:

```
SUSPENDED BY SYSTEM RSVP
```

When one or more active processes are suspended, any new user processes are scheduled. Refer to "Process Scheduling" earlier in this section for details.

Second Chance Overlay Mechanism (SCOM)

As described earlier in this section, the second chance overlay mechanism (SCOM) is a memory management procedure.

If the amount of available memory drops too low, the SCOM attempts to remedy the situation by overlaying regions of memory that have not been used recently. If this does not free sufficient memory, the SCOM suspends the lowest-priority process in the mix. If two processes are of equally low priority, the SCOM suspends the process that has been executing for the shorter amount of time.

The SCOM is initiated by setting the OLAYGOAL memory management parameter to a nonzero value. Use the SF (Set Factors) system command to assign a value to this parameter.

You can specify the amount of memory that must be kept available by using the SF command to assign a value to the AVAILMIN memory management parameter. The SCOM begins suspending processes when the total amount of available memory drops below one-half the value of AVAILMIN.

Invisible independent runners, MCSs, and control programs are never suspended for lack of memory. Also, processes that are using the MCP SORT facility cannot be suspended for lack of memory.

Suspending for Insufficient ASDs

If the number of available ASDs drops below 5 percent of the total number of ASDs, then the system suspends any process that needs to access more ASDs. Even a control program is suspended in this situation. When the number of available ASDs increases, the processes are automatically resumed, approximately in priority order. For an explanation of the ways to inspect or change the number of ASDs, refer to "Process Scheduling" earlier in this section.

Resuming Suspended Processes

Suspended processes appear in the Waiting Entries list with an RSVP message. A process is automatically resumed when the amount of available memory or ASDs increases.

Suspended processes can be resumed manually by

- Assigning a higher value to the PRIORITY task attribute of the process with the PR (Priority) system command. The system resumes any suspended process whose priority has been changed by this command.
- Entering an OK (Reactivate) system command in response to the RSVP message for a process.

Note, however, that a suspended process that has been resumed manually is subject to immediate suspension if additional memory or ASDs have not become available.

Resident Programs

If your site has programs that are run repeatedly (many times a day, for example), you might consider making them *resident programs*. A resident program initiates more quickly because the code segment dictionary remains in memory after the program has been run once. This causes decreased processor and I/O usage because the old code segment dictionary can be reused instead of a new one being created. However, resident programs occupy memory even when nobody is using them, resulting in less available memory for other processes.

Base your decision to give programs resident status on how your system is used and the programs you run. The more resident programs you have, the more likely it is that large processes will run into a shortage of memory. This is only a concern if you are running enough processes to approach the limit of the system's memory.

The RP (Resident Program) system command is used to mark a code file as resident, remove the resident status of a code file, and display all the resident programs on the system. You can use the STRUCTURECACHE (Cache Maintenance) system command to specify how long code segment dictionaries that are not resident programs remain in the system memory. More information about the RP system command and the STRUCTURECACHE system command can be found in the *System Commands Reference Manual.*

Control Programs

Giving *control program* status to a program places it in a higher priority category than is normally assigned to user programs. The control program category is the same category as for WFL jobs and MCSs, meaning that only the system software has a higher priority.

Marking a user program as a control program means that there is less likelihood that the process will be scheduled or suspended, but does not guarantee it. However, a control program rarely should be scheduled or suspended.

The primary reason for making a program a control program is to make it run faster. This can be very useful for interactive programs such as a text editor, which most often uses very little processor time but requires quick response time.

You should be careful not to indiscriminately designate programs as control programs. Control programs are immune to being scheduled because of insufficient available memory. If there is a shortage of available memory, then submitting an excessive number of control programs could result in severe thrashing. Also, if a bug in a control program causes it to loop, it could use all the available processor time and prevent any lower priority processes from being executed.

Programs are designated as control programs with the CONTROL option of the MP (Mark Program) system command. This command option does not affect any instances of the designated program that are already executing. The command also does not affect any processes whose code segment dictionaries were created before the MP command was entered. A process might be using an old code segment dictionary if the program was previously marked as a resident program or if the process shared a code segment dictionary with a process that was initiated earlier.

Note: *Use the CONTROL option with caution to set priorities above 80 since MCSs, data comm control stacks, and network control stacks all run in the same priority class as control programs. Setting a priority value higher than 80 for a control program could cause network support processors (NSPs) or network processors (NPs) to fail.*

For information about the MP system command, refer to the *System Commands Reference Manual.*

Section 10
Managing the System Workload

The way in which the operating system controls the system workload by using features such as scheduling and suspension is described in Section 9.

At the job and task level, you can control the workload through various system commands and message control system (MCS) control commands. These commands can have a profound effect both on overall system performance and on the performance accorded to different types of processes or different types of users. This section describes how you can use these commands and the various features of the Work Flow Language (WFL) to manage the workload.

Processes can be initiated using the following jobs, statements, and features:

- WFL jobs
- WFL statement from MARC or CANDE
- START statement from MARC or CANDE
- RUN statement from MARC or CANDE
- COMS window
- ??RUN command from an ODT
- CALL, PROCESS, RUN, or ZIP statement in a program

Processes that are initiated through a MARC or CANDE *RUN* command are not subject to control by the job queue mechanism. Thus you cannot control individual processes through MARC or CANDE as effectively as you can through WFL.

The capabilities of WFL are summarized in the *Task Management Guide*, and details about the language are contained in the *WFL Reference Manual.*

This section discusses the choices and controls available over process initiation through WFL, and through COMS, MARC and CANDE sessions. For detailed information on memory management, consult the *A Series Memory Subsystem Administration and Operations Guide.* An overview of system performance monitoring is provided in the *A Series Systems Functional Overview.*

Managing WFL Jobs

Work Flow Language (WFL) is a task-oriented job control programming language designed primarily for compiling and running programs written in other languages. Initiating processes through WFL jobs provides a flexible yet effective way to group programs together and control access to system resources.

When started, WFL jobs are placed in job queues that can be defined to have particular characteristics. After a WFL job has been compiled, it is placed in a particular queue according to its requirements and the definitions of the queues. Only WFL jobs go through the job queue mechanism. Most of the time they do so automatically, but in some situations a WFL job can circumvent the job queues. Refer to the *Task Management Guide* for more information about WFL jobs and their treatment by the system.

You can have the greatest control over system performance and process initiation by using job queues and establishing a policy for their use. Jobs that enter the mix through uncontrolled means can cause other processes to be scheduled and active processes to be suspended.

For instance, someone could enter a CANDE *RUN* command to initiate a compilation of a large program and set the priority of the job at the highest value. If this was done during a time when important production or interactive programs are running, it could steal processor time and memory away from those programs, causing response time to degrade or programs to be suspended.

Queues can be resource oriented. A site with few tape drives might want to use queues to ensure that jobs did not wait in the mix for tape drives.

Some important considerations regarding WFL jobs are

- The process types, if any, that you want to control through the job queues
- The typical resource usage pattern for your most important processes
- The response time goals for your most important processes
- Whether or not to allow multiple task initiation
- The policy for charging users for resources used

You might consider a policy where most processes are initiated through WFL jobs. The most useful would be to run all batch programs as WFL jobs. Interactive data-entry type programs such as a text editor would not benefit from being controlled through the job queues.

As you read this section, you should attempt to formulate a plan for the creation and institution of job queues. Some questions you should consider are

- How many job queues do I need?
- Do I need a default queue?
- In what order should I create my job queues?
- Do I need to prioritize the queues?
- What processes am I going to run that are more important than others?
- Do I need to set processor and I/O limits?

Job Queues

A job queue is a structure containing a list of jobs that are waiting to be initiated. An A Series system can have up to 100 job queues, but a job cannot be listed in more than one queue. The number of jobs that are already in a job queue does not affect the ability to add new jobs to that queue.

When the system is initialized with a cold-start, one job queue is automatically created and marked as the default queue. It is designated as job queue 0. Refer to "Default Job Queue" later in this section for more information. If no job queues exist (they all have been deleted), then WFL jobs cannot be initiated. The system discontinues each job and displays the following error message:

```
JOB DSED OUT OF QUEUE
```

Each job queue can be assigned queue attributes, and each job queue can have a different set of attributes or attribute values. Queue attributes regulate the types of service that are given to the jobs in a queue. By employing queue attributes, you can

- Specify limits on the resource usage of jobs initiated from a given job queue. These limits apply to the total resource usage of the job and all its tasks.
- Specify how many jobs and tasks from that queue can be executing at the same time. This number is called the *queue mix limit.*

A queue starts with no attributes except the queue number. As long as all job queues have no queue attributes assigned, only the CLASS job attribute ultimately prevents a job from being accepted by any particular queue.

Within each job queue, jobs are ordered based on their priority. When jobs have the same priority (referred to as the *priority class*), ordering is based on the order in which jobs were submitted. At any given time, only one job in each job queue is visible to the operating system as a candidate for initiation.

Multiple job queues can be desirable for the following reasons:

- To regulate the way system resources are divided among different groups of users and different types of jobs. For example, the job queue mechanism can be used to assign a higher priority to jobs that use little processor time. Multiple queues can thus improve service for short jobs.
- To improve overall system performance by preventing memory from becoming overcrowded. The job queue mechanism can be used to limit the number of WFL jobs in the mix. While jobs are waiting in job queues, they are stored on disk and do not take up any space in main memory.

The job queue mechanism offers another benefit that is available even if only a single job queue is used. This is the capability to schedule a job for execution at a later time or date. For example, a large job can be scheduled to run outside peak hours, thus reducing the load during the day. This capability is discussed under "Scheduling Job Initiation" later in this section.

Access to Job Queues

Access to job queues is controlled by three factors:

- Usercode attributes
- Job attributes
- Job queue attributes

These attributes all provide ways for you to control access to the job queues by users of your system.

Usercode Attributes

Usercode attributes can be thought of as control parameters that define what any particular usercode is allowed to do. There are three usercode attributes that relate to a user's access to the job queues:

Attribute Name	Description
CLASS	This attribute specifies the default queue to use for jobs initiated by this usercode. If no class is specified, the default queue is used unless the QFACTMATCHING algorithm (described later) is being used, in which case an appropriate queue is selected.
CLASSLIST	This attribute specifies the classes (queues) that the usercode is allowed to access. However, use of this attribute is determined by the value of the ANYOTHERCLASSOK usercode attribute.
ANYOTHERCLASSOK	When this attribute has a value of TRUE, the usercode is allowed to access any class (queue) except those specified by the CLASSLIST attribute. When this attribute is FALSE, the usercode can use only those classes specified by the CLASSLIST attribute.

Task Attributes

By using certain task attributes, the person who initiates a job can specify values that can affect the placement of the job in the job queues. Also, by setting certain task attribute values and initiating a job in one of the ways that avoids the job queues, a user on the system can bring jobs into the mix that you might not want there at certain times.

Job Queue Attributes

Queue attributes specify the characteristics of the job queues. With queue attributes you can create queues for handling different types of jobs. For example, one job queue can be established to accept jobs that do not require much processor time; another can accept longer jobs; others can limit I/O time or require that a job file exist on a particular family.

The available queue attributes are discussed later in this section.

Job Queue Order

Initially, the job queue order corresponds to the order in which the job queues were created. The lowest-ordered job queue is the one that was created first; the highest-ordered job queue is the one that was created last. However, if a job queue is deleted, then its place in the job queue order becomes open. The next job queue to be created takes that place in the job queue order.

The job queue order affects the following:

- The selection of a job queue for a job that has no CLASS specification. Refer to "QFACTMATCHING Algorithm" later in this section for details.
- The order in which queues are selected when jobs are to be started. Refer to "Selecting Jobs from Queues" later in this section for details.
- The order in which job queues are displayed by the ML (Mix Limit), QF (Queue Factors), and SQ (Show Queue) system commands.

If the PRIORITY value does not determine which of several jobs should be chosen, the job from the lowest-ordered queue is selected first. Remember, however, that depending on other characteristics of the job, the operating system might schedule such a job immediately and allow jobs selected after it to be processed.

The job queue order bears no relationship to the job queue numbers. Job queue 50 is not necessarily of a higher order than job queue 10.

You can display the current job queue order by using the ML (Mix Limit) system command. This command lists all the job queues in job queue order, starting with the lowest-ordered job queue.

Mix limits, FETCH statements, and start times are the only factors that will actually prevent the CONTROLLER from selecting jobs from the queues. (The CONTROLLER is the operating system independent runner that handles the job queue mechanism and most system commands.) The rest is left up to the operating system-level controls (scheduling and suspending).

QFACTMATCHING Algorithm

If the CONTROLLER is compiled with the QFACTMATCHING option set and a job has no CLASS assignment, then the QFACTMATCHING algorithm examines the job queues in job queue order, from the highest-ordered job queue to the lowest-ordered job queue. The system places the job in the first job queue that meets both of the following requirements:

- All resource limit attributes specified in the job attribute list must have values less than or equal to the corresponding resource limit attributes of the job queue. For example, if the job attribute list includes a statement that assigns the job a priority of 55 and the job queue has a priority limit of 50, then the system will not enter the job in that job queue.
- The job queue must be one that is valid for a job with this usercode.

If none of the job queues can accept the job, the system discontinues the job and displays the message "Q-DS." You can determine whether the QFACTMATCHING algorithm is set by using the CO (Compile-time Options) system command. Unisys currently supplies operating systems with the QFACTMATCHING option reset. If you wish to set this option, you will have to purchase the symbolic file for QFACTMATCHING from Unisys and recompile it.

Job Queue Verification

The system makes a preliminary job queue assignment based on one of the following methods: the CLASS attribute value, the default queue, the QFACTMATCHING algorithm, or an assignment from a particular input device through a UQ system command. The use of the UQ (Unit Queue) system command is described later in this section under "Unit Queue Associations." Before actually inserting the job into the chosen queue, the system performs some additional tests.

First, the system examines the resource limits assigned in the job attribute list and compares them with the resource limits specified for the job queue. The system allows the job into the job queue only if all the job's resource limits have values less than or equal to the values of the corresponding queue resource limits. (This is the same test that was applied to jobs in the QFACTMATCHING algorithm at an earlier stage.)

Next, the system compares the FAMILY task attribute of the job with the FAMILY attribute of the job queue to determine if the values are compatible. The job FAMILY and the queue FAMILY are considered compatible if any of the following three conditions are met:

- There is no FAMILY queue attribute defined for the job queue.
- The FAMILY task attribute of the job is null.
- The FAMILY task attribute value equals the FAMILY job queue value.

If the job FAMILY and the queue FAMILY are not compatible, the system discontinues the job.

If the job passes all these tests, the system inserts it into the requested job queue.

Default Job Queue

The system provides one job queue as the default queue when the system is first initialized: job queue 0. Note that job queue definitions are not affected by a halt/load. This queue has no queue attributes associated with it, but is sufficient to allow WFL jobs to be submitted. You can modify or delete this queue, or you can designate another default queue with the DQ system command. However, if no queues exist, no WFL jobs can be initiated. In that case, all WFL jobs are discontinued with the error message "JOB DSED OUT OF QUEUE," which in this case means that no acceptable queue exists for them to join.

Many jobs do not have a CLASS attribute value or a unit queue association to specify the job queue in which they should be placed. The system can use either of two different

methods to determine the job queue for such jobs. The CONTROLLER compile-time option QFACTMATCHING specifies which of these two different methods the system should use.

If the CONTROLLER is compiled with the QFACTMATCHING option set, then the system uses the QFACTMATCHING algorithm to determine the job queue for an unassigned job.

If the CONTROLLER is compiled with the QFACTMATCHING option reset, then the system usually places unassigned jobs into the default job queue. The DQ (Default Queue) system command allows you to specify which job queue is the default queue. This command also can be used to change the default queue or to remove the default queue setting altogether.

Even if QFACTMATCHING is reset, the system still uses the QFACTMATCHING algorithm in the following cases:

- If there is no default queue defined for the system, because you used the DQ command to remove the default queue setting.
- If the default queue is one that is not correct for a job with this usercode. The values of the CLASSLIST and ANYOTHERCLASSOK usercode attributes determine whether a particular job queue is valid for that usercode.

Job Queue Active Count

Each job queue has an active count, which varies over time. The active count measures the number of active processes that originated, directly or indirectly, from that queue. The active count is computed as the sum of

- The number of in-use WFL jobs that originated from that job queue. This includes any scheduled or suspended WFL jobs from that queue.
- The number of in-use tasks that are descendants of those WFL jobs. This includes any in-use tasks that are scheduled or suspended. However, each WFL job is allowed one in-use task free. (That is, if a job has four in-use tasks, only three affect the active count.) Also, if any of these WFL jobs or tasks initiates a job, the new job and its tasks are not included in the active count for the job queue.
- Any jobs in the job queue that are waiting for RESOURCE (usually tape) or FETCH specifications. These jobs are included in the active count the system uses, but are not included in the active counts displayed by the QF (Queue Factors) or ML (Mix Limit) system commands. Jobs that are waiting on a STARTTIME are not included in the active count.

The active count for a job queue can exceed the queue mix limit if a job from that job queue initiates multiple tasks or if the FS (Force Schedule) system command is used to force selection of a job regardless of the queue mix limit. For information about the FS command, refer to "Forcing Job Initiation" later in this section.

Defining Job Queues

Job queues are created, modified, and deleted with the MQ (Make or Modify Queue) system command. Each job queue is identified by a unique number. Job queue definitions are preserved across a halt/load.

Assigning Job Queue Attributes

Job queue attributes enable you to create queues that can accept jobs with specific characteristics and control the selection of jobs for processing. Some queue attributes can slow down or speed up the selection of jobs from a queue; other queue attributes can help keep the queue from being blocked by jobs that are suspended or looping indefinitely. For example, jobs that are suspended on a NO FILE condition are still included in the active count for the job queue. If the queue mix limit is small, suspended jobs can prevent any new jobs from being initiated.

Still other attributes can limit the usage of particular types of peripheral devices. These are especially useful when one type of device is in short supply.

The available job queue attributes are discussed on the following pages.

NUMBER Queue Attribute

The NUMBER queue attribute assigns an identification number to a job queue. This number cannot be modified once the queue has been created. The only way to get rid of a job queue number is to delete the queue.

The job queue identification number must be in the range 0 to 1023, inclusive. This number is used only to identify each queue; it has no effect on queue priority. However, some numbering scheme may benefit system users. For example, the numbers could be linked to the priority of the job queues, with higher-numbered job queues having a higher PRIORITY queue attribute. It is definitely a good idea to assign numbers in an ascending or descending order when you create the job queues because of the system's queue ordering. Refer to "Job Queue Order" earlier in this section.

MIXLIMIT Queue Attribute

The MIXLIMIT queue attribute prevents the selection of any jobs from the job queue if the active count equals or exceeds the value of MIXLIMIT. (This value is referred to as the queue mix limit.) MIXLIMIT is one of the most important and widely used queue attributes. While most of the other queue attributes are used to define classes of jobs, the MIXLIMIT queue attribute effectively limits the total resource usage of any particular class. This queue attribute can be used to improve service for all active processes by preventing system resources from being overloaded.

TURNAROUND Queue Attribute

The TURNAROUND queue attribute speeds the service time for a particular job queue. The turnaround time for a job queue is the time since the selection of the previous job

from that queue. The system selects jobs first from the queues whose turnaround time has exceeded the value of their TURNAROUND queue attribute.

Note that only the mix limit can override the turnaround time. For more information about the factors that affect job selection, refer to "Selecting Jobs from Queues" later in this section.

PRIORITY Queue Attribute

The PRIORITY queue attribute corresponds to the PRIORITY task attribute. The PRIORITY task attribute affects the share of processor time, memory space, and other resources received by the job and its tasks. For further discussion of this topic, refer to "Process Priority" in Section 9.

PROCESSTIME Queue Attribute

The PROCESSTIME queue attribute corresponds to the MAXPROCTIME task attribute. The MAXPROCTIME task attribute limits the total amount of processor time that can be used by the job and its tasks. The operating system terminates a job or a task that exceeds this limit and displays the message "EXC PROC TIME."

If a job or task is terminated for exceeding the processor time limit, the job should be evaluated to see if a programming error exists or if the job simply needs more time to run. If the job just needs more time, you should change the job's CLASS so it will run in a queue with a higher PROCESSTIME limit.

The main purpose of the PROCESSTIME queue attribute is to force users to place their jobs in the proper queue. For example, if there is a high-priority job queue that is intended for short jobs, then the PROCESSTIME queue attribute is used to prevent users from placing jobs in that queue that take a long time to execute.

Programmers should beware of placing their jobs in queues with a PROCESSTIME limit lower than the amount of processor time the job might use. A job can be 99 percent complete, but if it exceeds the PROCESSTIME limit, it is abnormally terminated and all the work might be lost.

IOTIME Queue Attribute

The IOTIME queue attribute corresponds to the MAXIOTIME task attribute. The MAXIOTIME task attribute limits the total amount of I/O time the job and its tasks can use. The operating system terminates a job or a task that exceeds this limit and displays the message "EXC I/O TIME."

If a job or task is terminated for exceeding the I/O time limit, the job should be evaluated to see if a programming error exists or if the job simply requires more I/O time. If the job just needs more I/O time, you can change the CLASS attribute for the job so it runs in a queue with a higher IOTIME limit.

WAITLIMIT Queue Attribute

The WAITLIMIT queue attribute corresponds to the WAITLIMIT task attribute. This attribute specifies the maximum time the job or its tasks can wait after executing a wait statement. The operating system terminates a job or a task that exceeds this limit and displays the message "WAIT TIME LIMIT EXCEEDED."

The WAITLIMIT attribute does not affect the length of time a process can remain suspended. For example, a process waiting on a NO FILE condition is not affected by the WAITLIMIT.

ELAPSEDLIMIT Queue Attribute

The ELAPSEDLIMIT queue attribute corresponds to the ELAPSEDLIMIT task attribute. This attribute specifies the maximum time that can pass between job initiation and job termination. The reference is to actual clock time rather than processor time. The operating system terminates a job or task that exceeds this limit and displays the message "ELAPSED TIME LIMIT EXCEEDED."

When using this queue attribute, remember that the elapsed time for a job can vary widely from one run to the next. Depending on how busy the system is, the job might spend greater or lesser amounts of time in the ready queue or waiting for peripherals. Also, the job might have to wait for an operator to load a tape or make some other response to an RSVP message. Thus, setting a low ELAPSEDLIMIT value might cause some jobs to be discontinued even though they are executing as intended.

DISKLIMIT Queue Attribute

The DISKLIMIT queue attribute corresponds to the DISKLIMIT task attribute. This attribute specifies the maximum number of segments of disk space that the job and its tasks can request. The operating system terminates a job or a task that exceeds this limit and displays the message "DISK LIMIT EXCEEDED" or a similar message.

TASKLIMIT Queue Attribute

The TASKLIMIT queue attribute corresponds to the task attribute of the same name. This attribute limits the total number of descendant tasks that a job can generate. This attribute accepts a single value; it does not have separate default and limit values.

A job can be accepted into a job queue having a TASKLIMIT value different from that of the job. However, the TASKLIMIT value of the job is changed to match that of the job queue. This change is made even if the job specified a lower TASKLIMIT than the job queue.

Once the job is initiated, the job can change its TASKLIMIT to a value higher or lower than the job queue TASKLIMIT. If the job exceeds its current TASKLIMIT value while it is executing, the system terminates the job and displays the message "TASKLIMIT EXCEEDED."

LINES Queue Attribute

The LINES queue attribute corresponds to the MAXLINES task attribute. This attribute specifies the maximum number of lines of printer output that the job and its tasks can produce. The operating system terminates a job or a task that exceeds this limit and displays the message "PRINT LIMIT EXCEEDED."

Note that this attribute is valid whether or not spooling is used.

CARDS Queue Attribute

The CARDS queue attribute corresponds to the MAXCARDS task attribute. This attribute specifies the maximum number of cards that the job and its tasks can punch. The operating system terminates a job or a task that exceeds this limit and displays the message "PUNCH LIMIT EXCEEDED."

Note that this attribute is valid whether or not spooling is used.

FAMILY Queue Attribute

The FAMILY queue attribute corresponds to the task attribute of the same name. This attribute specifies the families on which to create new files or search for existing files. A job can specify a value for FAMILY in the job attribute list; otherwise, the job inherits the FAMILY value associated with the usercode of the job.

A job cannot be accepted into a job queue having a FAMILY value different from that of the job. The system terminates a job that is placed in such a job queue and displays the error message "QUEUE FAMILY MISMATCH."

After initiation, the job can change its FAMILY to a value different from the job queue FAMILY or assign a different FAMILY value to a task.

Tape Specification

The queue definition can also include a tape specification, which indicates the maximum number of tape drives a job from this job queue is expected to use. The queue tape specification is used only when the system is deciding whether a particular job is allowed in this job queue. The job is not allowed to join the queue if the RESOURCE task attribute of the job specifies more tape drives than are permitted by the queue tape specification. (The RESOURCE task attribute has effects on job selection—refer to "Selecting Jobs from Queues" later in this section.) The queue tape specification cannot be assigned a default value; only a limit value can be set.

Defaults and Limits

A programmer can include a job attribute list at the start of a WFL job to specify values for these task attributes of the job. The values of these task attributes also can be changed by task assignment statements later in the job:

Queue Attributes	Resource-Limiting Task Attributes
PRIORITY	PRIORITY
IOTIME	MAXIOTIME
PROCESSTIME	MAXPROCTIME
LINES	MAXLINES
CARDS	MAXCARDS
WAITLIMIT	WAITLIMIT
DISKLIMIT	DISKLIMIT
ELAPSEDLIMIT	ELAPSEDLIMIT

Separate default and limit values can be specified for each of these queue attributes.

If a default value is assigned to one of these queue attributes, it establishes a default value for the corresponding task attribute of the jobs in that queue. This default value is applied unless the job attribute list specifies a value for that task attribute.

If a limit value is assigned to one of these queue attributes, it prevents jobs that request a higher limit from being accepted into the queue. It can also act as a default. For example, if the PRIORITY queue attribute is assigned a limit value of 50, then no job in that queue can run with a PRIORITY task attribute value higher than 50. A job that attempts to set its PRIORITY task attribute to a higher value will be unsuccessful–that is, the value will not be changed. The only way to override the queue priority is with the PR system command. For details about how these limits are applied, refer to "Queue Limit Enforcement" later in this section.

The system also uses the limit values of these queue attributes to determine whether a particular job can be entered into a particular queue. For example, if the job attribute list for a job specifies a higher PRIORITY than a certain job queue allows, then that job cannot be entered into that job queue. For further information, refer to "Selecting the Queue for a Job" later in this section.

The limits that are established apply to the accumulated resource usage of the job and all its descendant tasks. For information about how the appropriate task attribute values are propagated from a job to its tasks, refer to the *A Series Task Attributes Programming Reference Manual.*

Limiting the processor time that a job can use helps to ensure a high level of availability for system resources. These limits force users to put their jobs in the proper queue. However, note that jobs that exceed the imposed limit are discontinued and must be recovered.

Queue Limit Enforcement

The queue attributes that specify limits affect more than the initial values of the corresponding task attributes. For the most part, they also prevent the job from assigning higher values to these task attributes after the job is initiated.

For the PRIORITY, LINES, CARDS, and DISKLIMIT queue attributes, if a job attempts to assign the corresponding task attribute a value higher than the queue limit, the task attribute value remains unchanged. However, no error or warning message is displayed and the job proceeds normally. The system operator can use the PR (Priority) system command to assign a job a PRIORITY value that is higher than the job queue limit. This action overrides the corresponding queue attribute.

For the PROCESSTIME and IOTIME queue attributes, the values of the corresponding task attributes cannot be raised after job initiation. When an attempt is made to change the value of one of these task attributes, the new value that results is the lower of the following two values:

- The value requested
- The current value minus the current accumulated usage

Thus, an attempt to raise the value of one of these attributes can result in the value actually being lowered.

Limits set by the ELAPSEDLIMIT and WAITLIMIT queue attributes are not strictly enforced. The corresponding task attributes must have a value within the specified limit at job initiation time; however, the job can later assign values higher than the limit.

Likewise, once a job is initiated, a job can attempt to access more tape drives than were listed in the tape specification for a job queue. The tape specification is not intended to forcibly limit a job to using a certain number of tapes. Rather, the tape specification is meant to protect a job against deadlock situations by warning the system how many tapes will be needed by the job and its descendants. For further information, refer to the *Task Management Guide*.

Selecting the Queue for a Job

The selection of the job queue for a job is based on several factors. You can influence job queue selection by including appropriate task attribute assignments in the job attribute list. Various system commands and usercode attributes can influence job queue selection.

Requesting the Queue for a Job

The CLASS task attribute does not override the limits established for a usercode. Use the CLASSLIST attribute to control placement of jobs in classes. The task attribute can override the usercode attribute, but an assignment is made only if it is valid according to other specifications.

You can select the job queue for a particular job by including a CLASS task attribute assignment in the job attribute list. This task attribute stores the queue number of the requested job queue. If none is assigned, a queue is selected based on factors such as the default queue specification, the usercode definition, and resource limits set for the queues.

A default CLASS value can be assigned for each usercode through the MAKEUSER utility. The CLASS item in the usercode definition stores this default. The value of the CLASS item is inherited by a job started under that usercode unless the job attribute list supplies a different value.

Additionally, the MAKEUSER utility can be used to specify which job queues are allowed for a particular usercode. The CLASSLIST and ANYOTHERCLASSOK attributes in the usercode definition store this information.

If the CLASS task attribute of a job specifies a job queue that is not allowed for that usercode, then the system terminates the job at compilation time and displays the message "ERROR: ILLEGAL CLASS." The system also discontinues a job and displays a "Q-DS" message if the CLASS task attribute value specifies a job queue that does not exist.

Unit Queue Associations

The UQ (Unit Queue) system command causes all jobs submitted from a particular device to be inserted into a particular job queue. The device types that can be specified in this command are ODTs, card readers, and tape drives. For example, you can use this command to put all WFL jobs submitted from a particular ODT into a high-priority queue. At a site where card readers are used, this command can be used to force all jobs submitted at a particular card reader to go into the same queue. A unit queue specification for a tape drive affects jobs that are initiated from tapes by way of the LD (Load Control Deck) system command.

The setting of the UQ command provides more than a default; it absolutely limits the choice of a job queue for jobs submitted from a particular device. If the requested job queue does not exist, or if the CLASS task attribute requests a different job queue, then the system discontinues the job and displays the message "Q-DS."

Requeuing Jobs

If the MQ system command is used to change the attributes of a queue, then the system reexamines any queued jobs to determine if they still qualify for that job queue. If not, the jobs are redistributed to other job queues. Also, if an MQ command deletes a job queue, the jobs are redistributed to other job queues. If no other suitable queue exists for a redistributed job, that job is discontinued.

However, once the operating system selects a job for initiation, it is not affected by changes in the queue definition. The only exception to this rule is if a halt/load occurs while the job is executing. After the halt/load, the job is automatically requeued so that it can be restarted. If the MQ command was used to change the queue definitions after the job was initiated, but before the halt/load, then the job may no longer qualify for the same queue when it is restarted. Once a job is requeued, it can be affected by system commands such as PQ (Purge Queue), FS (Force Schedule), and PR (Priority).

Ordering Jobs in a Queue

The jobs in a job queue are ordered according to priority, with the highest-priority job positioned at the head of the job queue. The relative priority of the jobs is determined by the value of the PRIORITY task attribute for each job. For information about how this task attribute receives its value, refer to "PRIORITY Task Attribute" in Section 9.

Jobs that have the same priority are ordered according to how long they have been waiting in the job queue, with the older jobs being placed nearer the head of the job queue. Note that the TURNAROUND queue attribute makes a comparison between the lead jobs in different queues, not between jobs in the same queue.

Changing the Job Order of a Queue

The position of jobs in a job queue can be changed with either the MOVE (Move Job/Pack) or the PR (Priority) system command.

A halt/load causes the jobs in a job queue to be restored to the standard order described earlier under "Ordering Jobs in a Queue."

Changing the Job Priority

The PR (Priority) system command changes the value of the PRIORITY task attribute. When the PRIORITY value of a job changes, the system reevaluates the position of the job in the queue. Thus, for example, you can advance a job to the head of the job queue by assigning it a priority higher than that of any other job in that job queue.

Changing the PRIORITY of a job while it is in a job queue results in the new PRIORITY being used when the job is run. It is not possible to set the PRIORITY of a queued job to a value higher than the PRIORITY limit defined for that job queue. However, after the job exits the queue, its priority value can be changed with the PR command. If a halt/load occurs, the job is restored to its original priority.

Selecting Jobs from Queues

The CONTROLLER invisible independent runner selects jobs from job queues and presents them to the operating system for initiation. At any given time, each job at the head of a job queue is a potential candidate for initiation.

However, the CONTROLLER can delay selection of any or all of these jobs. Various system commands control several factors that affect job selection.

For purposes of job selection, the head job of a job queue is defined as the first job in the job queue that is not waiting on a STARTTIME, FETCH, or RESOURCE specification. Jobs that are waiting on FETCH or RESOURCE specifications are included in the active count for the job queue; jobs waiting on a STARTTIME specification do not affect the active count.

The ML (Mix Limit) system command specifies an overall mix limit for the system. The system's mix limit can be set to a value that is lower than the sum of all the job queue mix limits.

When deciding whether to select a job from the job queues, the CONTROLLER first checks to see whether the overall active count for the system equals or exceeds the overall mix limit. The overall active count is the sum of the active counts of all the job queues. No jobs can be selected while the overall active count equals or exceeds the overall mix limit.

Next, the CONTROLLER examines the job queues and determines which queues have active counts that are less than their queue mix limit. No jobs can be selected from a queue while the active count equals or exceeds the queue mix limit. The queue mix limit is set with the MQ system command, which assigns a value to the MIXLIMIT attribute of the job queue.

Next, the CONTROLLER checks to see if any of the eligible job queues have exceeded their turnaround time. If so, the CONTROLLER determines which of these job queues has exceeded the turnaround time by the greatest amount and selects the head job from that queue. The queue turnaround time is set with the MQ system command, which assigns a value to the TURNAROUND attribute of the job queue.

If none of the eligible job queues have exceeded their turnaround times, then the CONTROLLER examines the PRIORITY task attribute of the head job in each job queue. The highest-priority job is selected. For information about how the PRIORITY task attribute receives its value, refer to "PRIORITY Task Attribute" in Section 9.

If the CONTROLLER has to decide between more than one job having the same PRIORITY value, then the job is selected from the lowest-ordered job queue. For information about job queue order, refer to "Job Queue Order" earlier in this section.

Once a job is selected from a job queue for initiation, it becomes subject to the operating system workload management controls and can be scheduled or suspended by the operating system, if necessary. For more information, refer to Section 9, "Managing Processes at the Operating System Level."

If the head job in a job queue has a RESOURCE specification, then the job is not selected from the job queue until the specified number of tape drives is made available. As long as the head job of a job queue is waiting on a RESOURCE specification, no jobs can be selected from that job queue. For more information on this topic, refer to the *Task Management Guide.*

Scheduling Job Initiation

The STARTTIME specification provides a convenient means of scheduling a job for a time when the system load is low, such as in the evening or during a weekend. STARTTIME is also a convenient means of scheduling jobs that must run at regular intervals, such as every morning. For an example of a job that restarts itself every day, refer to "Using SMFII for Billing" in Section 12, "Billing for Use of System Resources."

The STARTTIME task attribute specifies the earliest time and date that a particular job can be selected from a job queue. This task attribute can be assigned only to WFL jobs. It can be assigned in the task attribute list of the WFL job or in the statement that initiates the WFL job. The STARTTIME (Start Time) system command and the CANDE *?STARTTIME* command also assign this attribute to a job in a job queue. However, any changes made using these commands are not maintained across a halt/load.

When a job is initiated with a STARTTIME specification, it is compiled immediately and placed in an appropriate job queue. The position of the job in the queue is determined by the priority and age of all the jobs in the queue, as discussed under "Ordering Jobs in a Queue" earlier in this section. The display produced by the SQ (Show Queue) system command includes information about the STARTTIME specifications of the jobs in a queue.

Situations can occur where there are one or more jobs with STARTTIME specifications at the head of a job queue. When the CONTROLLER selects a job from a particular job queue, it selects the first job in the job queue that has a STARTTIME value less than or equal to the current time or that has no STARTTIME value.

Preventing Job Initiation

The HS (Hold Schedule) system command prevents the initiation of new processes on the system, except for invisible independent runners. This command has the following effect on WFL jobs:

- If a job is started after the HS command is issued, the job is compiled and entered into a queue.
- While the HS command is in effect, WFL jobs continue to be selected from the queues for execution. However, the WFL jobs are scheduled as soon as they enter the mix and cannot complete initiation until you enter an *HS-* command or force them individually into the mix by using the FS (Force Schedule) system command. (If you suspect that a job is causing a system problem, you can use this method to isolate the offending job.)
- If a job is already executing when the HS command is entered, the job continues running normally. However, the job is unable to initiate any new tasks until you enter *HS-*.

Initiation of processes resumes when you use the *HS-* command.

Forcing Job Initiation

To force the initiation of a particular job that is waiting in a queue, you can use the FS (Force Schedule) system command. Normal job selection factors such as the queue mix limit, the overall mix limit, and the HS (Hold Schedule) command are ignored. The FS command can be applied to any job in a job queue, not just the first job.

If the active count for a job queue exceeds the queue mix limit, this condition might indicate that an FS command was used to force initiation of a job in that job queue.

However, the active count can also exceed the mix limit if any of the active jobs from that job queue have initiated multiple tasks.

Managing Processes Initiated by Sessions

You can impose some general limits on the number of processes initiated through MARC or CANDE sessions. For information about the process initiation commands available in MARC and CANDE sessions, refer to the *A Series Menu-Assisted Resource Control (MARC) Operations Guide* and the *A Series CANDE Operations Reference Manual.*

Communications Management System (COMS)

The Communications Management System (COMS) provides some indirect methods for limiting the number of processes a COMS user can run simultaneously and the total number of processes that can be submitted from COMS sessions at a given time. Note the following:

- Strictly speaking, there is no such thing as a COMS session. However, there are MARC sessions, and MARC can be accessed only through COMS. Additionally, other message control systems (MCSs), such as CANDE, can be accessed either through COMS or separately from COMS.
- The COMS windowing capabilities enable a user at a single terminal to have many MARC and CANDE sessions in progress at the same time. For example, whenever a user opens a new CANDE window dialog, COMS creates a new CANDE session, which is accessed through that dialog. By limiting the number of window dialogs that are allowed to a user, you also indirectly limit the number of processes the user can run simultaneously.

You can impose these limits by using the COMS Utility to make changes to the COMS configuration file. You can specify the number of window dialogs allowed for each user by using the following two COMS Utility menus:

- Window Activity

  This menu can be used to define a window that accesses an MCS. In the Maximum Dialogs field, you can specify the maximum number of dialogs that can be opened for this window at any station. You can define multiple windows that access the same MCS, and each of these windows can have a different Maximum Dialogs setting.

- Usercode Activity

  This menu defines the capabilities that are allowed for a particular usercode. You can use the Valid Window List field to list the windows that the current user is allowed to access. Thus, some users could be allowed to access a CANDE window that allows multiple dialogs, whereas other users could be limited to a different CANDE window that allows only one dialog per station.

Windows can be created that access any type of interactive program, not only an MCS. Under COMS, a program can be accessed through either a direct window, from which the program can route messages directly to COMS, or through a remote-file window. You must define these programs to COMS by way of the Program Activity menu. This menu can be used to set values for the following task attributes of a direct program (a program running in a direct window):

- CHARGE
- FAMILY
- PRIORITY
- USERCODE

Of these task attributes, PRIORITY is particularly important to workload management because it influences the amount of processor time and other system resources that are allotted to the direct program. For more information about priority, refer to "Process Priority" in Section 9.

In COMS you can specify the maximum number of stations that can access a window at one time; this maximum is specified in the Maximum Users field of the Window Activity menu. Additionally, the Maximum Copies field on the Program Activity menu can be used to regulate the number of copies of a program that can be active at the same time. Limiting the copies of a program helps to limit the share of system resources devoted to that program.

For a complete discussion of the uses of the COMS Utility, including the menus mentioned previously, refer to the *COMS Configuration Guide.*

You can enter a CANDE session without going through COMS. For example, entering *?MCS SYSTEM/CANDE* on dialog 1 of the MARC window causes the station to be released from COMS and placed entirely under the control of CANDE. Limits set in COMS do not affect any sessions that are not accessed through COMS. However, if COMS is not being used at a station, then that station can run only one session in any case because multiple window and dialog capabilities are available only through COMS.

Menu-Assisted Resource Control (MARC)

Each MARC session runs only one task at a time. The user cannot submit any further tasks until the original task terminates. Because of this, the COMS limit on the number of dialogs available to a user also limits the number of tasks that a user can initiate directly in MARC.

However, there is no limit to the number of WFL jobs that can be initiated from the same MARC session. Also, any WFL job or dependent process initiated from a MARC session can itself initiate any number of other processes. No means is available to limit the number of these indirect descendants of a MARC session. There is also no way to limit the resource usage of processes descended from MARC sessions.

Command and Edit (CANDE)

CANDE provides control commands that can limit the number of CANDE sessions and the total number of tasks run by CANDE sessions. These control commands are described in detail in the *A Series CANDE Configuration Reference Manual.* The following paragraphs provide an overview of these features.

Limiting the Number of Tasks

The CANDE *?MAXTASKS* control command limits the total number of user tasks that can run in CANDE sessions at the same time. If a user submits a task and the number of active CANDE tasks is presently greater than or equal to the MAXTASKS value, the following message is displayed:

```
#WAITING FOR AVAILABLE TASK
```

CANDE processes the task when the number of active tasks becomes lower than the value of MAXTASKS. The value of MAXTASKS is preserved over a halt/load.

The count of active CANDE tasks includes processes initiated by any of the following commands:

- ADD
- BACK
- COMPILE
- COPY
- DCSTATUS
- LFILES
- LOG
- PRINT
- RESEQ (when used to resequence a BASIC file)
- RUN (or its synonym EXECUTE)
- UTILITY
- WRITE

Additionally, if a CANDE *START* command is used to initiate a WFL job, the job is included in the count of active CANDE tasks until it enters a job queue. If a CANDE *WFL* command is used to initiate a WFL job, the job is included in the count of active CANDE tasks until it terminates. All WFL jobs go through the job queue mechanism and are affected by the controls discussed under "Managing WFL Jobs" in this section.

Only the immediate offspring of a CANDE session are included in the MAXTASKS count. For example, if the CANDE *RUN* command is used to initiate a program called PROG, the MAXTASKS count increases by one. But if PROG initiates three other programs, the MAXTASKS count is not affected.

The CANDE *?COMPILESPERCENT* control command defines the percentage of the maximum number of CANDE tasks that can be compiler tasks. For example, if MAXTASKS is 50 and COMPILESPERCENT is 50, then the maximum number of compiler tasks allowed at one time is 25. CANDE delays the initiation of a compiler task and displays the following message if the total number of active compiler tasks exceeds or equals the value specified by COMPILESPERCENT:

```
#WAITING FOR AVAILABLE COMPILE TASK
```

Limiting the Number of Sessions

Another method of indirectly limiting the tasks submitted from CANDE sessions is to limit the number of CANDE sessions that can be active at the same time. Each session can have only one task active at a time; therefore, limiting the number of sessions limits the number of tasks.

CANDE sessions come in two types: interactive sessions and schedule sessions. These two types of sessions are described in the *Task Management Guide.* Separate controls are used to limit these two types of sessions.

You can limit the total number of interactive CANDE sessions by using the *?MAXSTATIONS* CANDE control command. The value of MAXSTATIONS is the maximum number of stations that can be attached to CANDE at one time. Each CANDE dialog opened through COMS is counted as a separate station.

You can limit the total number of CANDE schedule sessions by using the *?SCHEDULE <limit>* CANDE control command. You can limit the total number of schedule sessions allowed for any one user by using the *?SCHEDULE USERLIMIT <user limit>* control command. If the number of schedule sessions equals or exceeds the limit, then any new schedule sessions are queued for later execution.

Section 11
Logging System Events

One of your most important tasks as the system administrator is to maintain a record of the actions that take place on the system. There are several reasons to keep a record of the system's actions, such as tracking security violations, maintaining billing records, and performing system maintenance.

Actions that take place on the system are recorded in a log file. The operating system creates and maintains three types of log file:

- The system log. This file stores information about various events that occur on the system. The file is titled SYSTEM/SUMLOG.
- Job logs. A separate log is created for each job on the system and stored in a file associated with the job. This job log contains information about the job and its descendant tasks.
- The system fault log. The fault log is a file of hardware error reports. The fault log contains information about peripheral errors that have occurred on your system. The file is titled SYSTEM/FAULTLOG.

The system log information is useful for diagnosing system performance problems, helping enforce security restrictions, or as input to a billing system. You might find it useful to consult the system log to retrieve information about specific jobs and tasks, or to examine usage information for specific devices.

Numerous types of information can be recorded in the system log. You can choose whether to log every action that occurs on the system or to log only particular types of actions. With a privileged usercode, several system commands can be used to specify the types of actions to be recorded in the system log and job logs. By restricting log recording to fewer types of actions, you can reduce system overhead and improve performance. You can also specify that actions associated with a particular usercode are to be logged.

Some of the types of information that can be logged in the system log are outlined in the following list:

- Information about all jobs, tasks, and sessions on the system, including their resource usage and types of terminations
- Information related to data comm, message control system (MCS), and BNA activity
- Information related to the hardware and software configurations of the system as well as to problems encountered during operation

Information from the system log can be extracted and analyzed by using any of the three available log analysis utilities or by using your own programs.

Depending on the value of relevant task attributes for the job, a copy of the job log is printed when the job terminates. The printout of the job log is called a *job summary*; it provides information about the history of a job and its tasks.

The remainder of this section provides an overview of the location and general format of the system log and the utilities that are available for extracting and analyzing system log information. For more information about job summaries, refer to the *Task Management Guide*. For more information about selective logging capabilities, refer to the *System Software Support Reference Manual*.

System Log File

The system log records actions that occur on the system in chronological order. There are many different types of log entries, each of which store different types of information. The various types of log entries are identified by major type and minor type numbers. For example, all log entries of major type 1 store information about jobs or tasks. Entries of major type 1, minor type 3 store information specifically about task initiations.

Each log entry includes a general part that identifies the major and minor type of the log entry. The remainder of the entry contains fixed-format data and variable-length data. For descriptions of the formats of all the types of log entries, refer to the *System Software Support Reference Manual*.

The system log is a disk file of type DATA. The current system log is titled *SYSTEM/SUMLOG. You can specify the disk family on which the system log is to be located by using the DL (Disk Location) system command.

Log records are added to the system log file until it contains approximately 95,000 records. At this point, the operating system releases the log. Releasing the log consists of creating a new system log and retitling the old one. The old log file is titled as follows:

```
SUMLOG/<system serial #>/<date>/<log number>
```

The date is a 6-digit integer having the form *mmddyy*. The log number is a 6-digit integer that is one greater than that assigned to the last system log that was released. The sequence of log numbers that is assigned to successive logs is not disturbed by a halt/load.

You can release the system log before it reaches 95,000 records by using the TL (Transfer Log) system command. Doing this allows you to maintain log files that contain records from specific time periods. For instance, you might want log files on a daily basis, or you might want to isolate log entries when running system tests.

Selective Logging

The LOGGING (Logging Options) system command can be used to modify the system log parameters. You can use this command to cause additional major or minor types to be logged, or to suppress logging of specific major or minor types. The changes made by

this command are retained across a halt/load. MARC provides access to this command by way of the LOG menu.

Unisys recommends that you always log the following events:

Event	Major Type	Minor Type
Establish Identity (EI)	0	1
Beginning of job (BOJ)	1	1
End of job (EOJ)	1	2
Beginning of task (BOT)	1	3
End of task (EOT)	1	4
All maintenance records	2	
MCS session log on (LOGON)	4	1
MCS session log off (LOGOFF)	4	2
Halt/load entry	6	1
SETSTATUS entry	6	3

The LG (Log for Mix Number) system command can be used to force the logging of all activities associated with specified mix numbers. All major and minor types of entries are logged for the specified mix numbers. This command enables you to track the activity of particular active jobs, tasks, and sessions. The values set by this command are not retained across a halt/load.

Selective Logging by Usercode

For any particular usercode, the types of entries that are logged can be controlled by using the MAKEUSER utility to assign a LOGSELECT usercode attribute to the definition of the usercode.

This usercode attribute can be used to enable logging of any combination of the following categories of activity associated with the usercode:

- Successful actions
- Failed actions
- Security relevant actions
- Security violation actions

If all these categories of logging are enabled, then all actions associated with the usercode are logged. A particular action associated with a usercode is logged if it meets the selection criteria in the LOGSELECT usercode attribute or the selection criteria in the system logging parameters, or both.

Changes made to a LOGSELECT usercode attribute do not affect processes that are already in progress. However, changes made to a LOGSELECT usercode attribute are

applied to subsequent tasks initiated from a session. To cause logging of a process that is already in progress, use the LG command.

For a detailed discussion of the LOGSELECT usercode attribute, refer to the *Security Administration Guide.*

Programmatically Controlled Selective Logging

A privileged DCALGOL process can set the system logging parameters by using the SETSTATUS intrinsic routine. There are SETSTATUS calls corresponding to the LOGGING and LG system commands.

The SETSTATUS version of the LG call has the same effect as that of the LG system command. However, the SETSTATUS version of the LOGGING call has greater capabilities than the LOGGING system command. These additional capabilities include the following:

- For each major and minor type to be logged, the call can further restrict logging to only those entries that fall into selected logging categories. These logging categories are successful actions, failed actions, security relevant actions, and security violation actions.
- The call can assign logging parameters for the system log, the job logs, or both. The parameters assigned for the system log can be different from the parameters assigned for the job logs.

For details about these calls, refer to the *GETSTATUS/SETSTATUS Reference Manual.*

Site-Defined Log Entries

In the operating system code file there are logical definitions of various types of log entries that are not normally used. A privileged program or MCS can insert log entries of any of these types into the system log by linking to a system library called MCPSUPPORT and calling an exported procedure called MCP_LOGGER.

For details about this capability, refer to the *System Software Support Reference Manual.*

Adding Comments to the Log

Comments can be entered into the system log with either of two system commands.

The LC (Log Comment) system command allows a string of text to be entered into the system log. The log entry containing this text will appear in the log in chronological sequence with the other log entries.

The LJ (Log to Job) system command allows a comment for a specific process to be entered into the log. The comment appears both in the system log and in the job log for that process, in chronological sequence with the other log entries.

Comments entered into a log with the LC or LJ command can be printed by running the LOGANALYZER utility with the OPERATOR option set. This utility is discussed in the *System Software Support Reference Manual.*

Periodic Logging

The PLI (Periodic Logging Interval) system command causes logging of usage information for open files at regular time intervals. Logging this information periodically can be useful because otherwise the accumulated usage information for a file is lost across a halt/load. This information can be used, for example, by a billing system to improve the accuracy of file usage charges.

Periodic logging causes a File Interval Record (Major Type 1, Minor Type 10) to be logged for each open file on the system the first time the file is accessed after the interval has passed. The length of the time interval is specified by the PLI command.

The File Interval Record includes information about the accumulated usage of a file. This information is also included in the File Close Entry (Major Type 1, Minor Type 6) that is logged when the file is closed. However, if a halt/load occurs between the time a file is opened and the time it is closed, the usage information in the File Close Entry reflects only usage since the halt/load.

This command should be used with caution because periodic logging impairs system performance. This effect is most severe when the logging interval chosen is very short.

Analyzing the Log

Unisys supplies the following utilities for use in extracting and analyzing information from system logs:

- LOGANALYZER
- LOGGER
- System Management Facility II (SMFII)

Each of these utilities can extract information from the log efficiently. However, they differ in their abilities to summarize and manipulate log information. SMFII provides the most sophisticated capabilities in this area.

Additionally, user-written programs can extract information from the log. The confidentiality of security-related log records can be enforced by setting up the system log so that user programs must use the SDASUPPORT library to access the system log. For more information, refer to "Using Programs to Access the System Log" later in this section.

LOGANALYZER Utility

The LOGANALYZER utility produces a report consisting of all system log entries that correspond to parameters that you set. LOGANALYZER does not perform any

processing on the information returned, it simply enables you (or any user) to inspect the log entries relevant to a particular subject.

Any user can initiate LOGANALYZER from a CANDE or MARC session or from a WFL job or other user program. LOGANALYZER can extract information from the current system log or from previous system log files.

The parameters used to select log entries include configuration and maintenance options, data comm options, and job, task, or session options. The job, task, or session options can extract any of several types of entries, including beginning of task, end of task, compilation error, and job entries. LOGANALYZER also can be used to extract only those records related to a process with a particular name or mix number.

For a complete description of this utility, refer to the *System Software Support Reference Manual.*

LOGGER Utility

The LOGGER utility produces a report containing various types of log information specified by parameters that you can set. LOGGER can present totals or averages of selected types of values, or it can list values of individual log entry items. LOGGER can also produce reports that present several types of data in tabular form.

Any user can initiate LOGGER from CANDE or MARC, or from a WFL job or other user program. LOGGER can extract information from the current system log or from previous system log files. LOGGER also can be used to produce reports covering an extended time period or summarizing information for the year to date.

LOGGER runs in two phases. In the first phase, it produces files called JOBSUMMARY and STATISTICS, and optionally, a file called FILEIODATA and a file called DRCDATA. The JOBSUMMARY file contains information related to jobs, tasks, and sessions. The STATISTICS file contains summary information about system usage. The FILEIODATA file contains information about file access actions. The DRCDATA file contains information on the amount of disk space used by usercode and by family, and, optionally, the amount of disk space that can be assigned to a user. The system must be running with the Disk Resource Control (DRC) system to create the DRCDATA file.

In the second phase, LOGGER extracts information from one or more of these files and produces a report.

For a complete description of this utility, refer to the *System Software Support Reference Manual.* An example of using LOGGER to produce a billing report is given under "Using LOGGER for Billing" in Section 12 of this guide.

System Management Facility II (SMFII)

SMFII produces a report containing detailed analyses of information contained in the system log file. The programs provide flexible control over the type of analysis to be performed, the logic used to select items from the log, and the formatting of the report.

SMFII is a collection of several utilities. The ones used for log analysis are the LOGCONSOLIDATOR program and the QUERY program.

The first step of log analysis is to run the LOGCONSOLIDATOR program, which analyzes one or more system log files and produces various database files as output. For purposes of log analysis, you should specify that only the database files belonging to the workload module are produced. These database files contain many types of data items. For descriptions of these data items, refer to the *SMFII Resource Management Reference Manual.*

The next step of log analysis is to run the QUERY program, which analyzes the workload module files and produces a report. You can display the report on the terminal screen or print it out.

The commands that cause QUERY to produce a report can be entered interactively. However, if you want a complex report, it is usually easier to store the report commands in a disk file. You can use the ENTER command to cause QUERY to read commands from the disk file.

One of the most important QUERY commands is the DEFINE command, which allows you to create commands that combine many other QUERY commands. Defines can be created for any desired type of report and reused later.

Unisys provides files containing sample defines that you can tailor to meet your site's needs. The sample defines file for analyzing the workload module is titled DEFINES/SMFII/WORKLOAD/STANDARD. Descriptions of these sample defines are included in the *SMFII Resource Management Reference Manual.*

SMFII can generate reports that analyze log data accumulated over a long period of time. This can be done in either of two ways:

- By saving the system log files. You can run LOGCONSOLIDATOR to produce a workload module based on multiple system log files.
- By combining multiple workload module files using the SMFII File Maintenance Utility. You can use the QUERY program to analyze the resulting workload module.

For more information about SMFII, refer to the *SMFII Resource Management Reference Manual* and the *SMFII Query Operations Guide.* An example of using SMFII to produce a billing report is given under "Using SMFII for Billing" in Section 12.

Using Programs to Access the System Log

Any user can write a program that reads information from system log files. Unisys provides a library called SDASUPPORT, which is a programmatic interface to system log files. A process can read the system log file directly or by means of the SDASUPPORT library.

The ability of a process to read from the system log depends to some extent on whether the process has *direct read access* to the system log file. A process receives direct read access if at least one of the following conditions is true:

- The process is privileged. For a description of the factors that determine process privilege, refer to "Privileged Program Status" in Section 3.
- The system log file has a SECURITYTYPE file attribute value of PUBLIC.
- The system log file has a SECURITYTYPE file attribute value of GUARDED or CONTROLLED, and has an associated guard file that allows read access to the process.

The SECURITYTYPE value of the system log file is inherited from the old log file when a new log file is created. The default SECURITYTYPE value is specified by the NONUSERFILES system security option, which can have a value of PUBLIC or PRIVATE. You can use the SECOPT (Security Options) system command to assign system security options such as NONUSERFILES.

The SDASUPPORT library exports procedures that perform such functions as opening or closing a system log and reading selected entries from a system log. A user program can link to the SDASUPPORT library by way of the function name SDASUPPORT.

On a system where InfoGuard security enhancement software is *not* installed, a process that has direct read access can read the log file directly or by means of the SDASUPPORT library. A process that does not have direct read access cannot read the system log by any means.

On a system where InfoGuard is installed, a process that has direct read access can read the log file directly or by means of the SDASUPPORT library. A process without direct read access cannot read the log file directly, but can use the SDASUPPORT library.

The version of SDASUPPORT used on InfoGuard systems is enhanced to act as a security filter for the system log. When a process reads the system log through SDASUPPORT, the system determines whether the process has direct read access to the system log, and takes the following actions:

- If the process has direct read access, then SDASUPPORT allows the process to read all the log entries.
- If the process does not have direct read access, then SDASUPPORT filters the log so that only some of the log entries can be read by the process. This filtering is invisible to the process, which views a virtual filtered system log file. SDASUPPORT uses the record visibility value in each log entry to determine whether to allow the process to read that log entry.

LOGGER, LOGANALYZER, and LOGCONSOLIDATOR all use SDASUPPORT as their interface to the system log. SDASUPPORT evaluates these programs for direct read access at run time, using the rules previously described. On a system that is not running InfoGuard, these programs can access the system log only if they run with direct read access rights. On an InfoGuard system, SDASUPPORT filters the log if the programs do not receive direct access rights.

For more information about SDASUPPORT, refer to the *System Software Support Reference Manual.* For more information about security, refer to the *A Series Security Features Operations and Programming Guide* and the *Security Administration Guide.*

System Fault Log

To help you track errors that occur in your system hardware, Unisys provides a system library called SYSTEM/FAULTLOGSUPPORT. This library creates a fixed size log file of hardware error reports, named SYSTEM/FAULTLOG, on the system's halt/load unit. This fault log contains summary information about peripheral errors which have occurred on your system; on A16 systems, it also contains information about any mainframe errors that have occurred.

The fault log is intended to enable your customer support engineer (CSE) to better maintain your system; in many cases it eliminates the requirement to analyze the system SUMLOG files when determining what is wrong with a system. You use the SL (Support Library) system command to link the FAULTLOGSUPPORT function name to the SYSTEM/FAULTLOGSUPPORT library to enable creation of the fault log. A Unisys proprietary analysis program to analyze the fault log, SYSTEM/FAULTLOGANALYZER, is available to customer support engineers.

For instructions on using the SL (Support Library) system command, see the *System Commands Reference Manual.*

Section 12
Billing for Use of System Resources

Often a system is used by several different users or groups of users who are billed separately for their usage of the system. The data used for billing can be generated programmatically through analysis of the system log. Because the billing requirements of each site vary so widely, Unisys does not supply a billing utility. However, Unisys does provide interfaces and functions that you can use in the development of a billing utility at your site. This section describes the billing-related features of A Series software.

Two basic strategies for billing are *historical billing* and *real-time billing*. Historical billing systems analyze one or more system log files and produce a summary of the usage and charges for the various users of the system. Real-time billing is available to CANDE, MARC, the operating system, and JOBFORMATTER using log record and task information to generate billing reports and summaries.

Historical billing is usually accomplished by one of the following methods:

- Using System Management Facility II (SMFII). With this method, you are responsible for producing a set of billing defines for use by the SMFII QUERY program. Designed primarily as a performance-analysis tool, SMFII has very sophisticated report-generation features.
- Using the LOGGER utility. With this method, you are responsible for producing a BILLINGSUPPORT library for use by LOGGER. Unisys provides a sample library that explains the input parameters that are used.
- Using a user program to extract billing information from the system log. Any user with programming expertise can write such a program. Note that the program might be required to use the SDASUPPORT library as the interface to the log. Refer to "Using Programs to Access the System Log" in Section 11 for further information.

Real-time billing can be accomplished through the second of the three methods used for historical billing, namely the use of LOGGER. Again you are responsible for providing an appropriate BILLINGSUPPORT library.

Real-time billing is controlled by the value of the BILLING_OPTIONS word in the BILLINGSUPPORT library. The system administrator can use this mechanism to select when billings should be processed. Bill processing may be provided in any combination of the following conditions:

- The operating system may invoke the billing library at BOT or EOT.
- JOBFORMATTER may request billing each time a job terminates.
- CANDE and MARC may request bill processing at the beginning and end of sessions (BOS/EOS), the beginning and end of tasks (BOT/EOT), and whenever status inquiries are made (for example, ?STA for CANDE users).

Note that in any case, billing is not performed until the BILLINGSUPPORT library has finished initialization and the operating system is linked to it.

The following pages review each of these billing strategies in more detail.

Using Chargecodes

Each process has a task attribute, called CHARGE, that is used specifically for providing billing information. This task attribute is provided as a means of indicating the user or group of users to which a process is to be billed. The value of this attribute is recorded in the system log entry for each process and thus can be accessed by a historical billing system that analyzes the system log. This value may also be accessed by a real-time billing system.

Generally, you define the chargecodes in use at a site and implement them by using the MAKEUSER utility. This utility can be used to define the following charge-related usercode attributes for each usercode on the system:

- CHARGEREQ

  If this item is set, then any session or process running under this usercode must have one of the chargecodes listed in the CHARGECODE item.

- CHARGECODE

  This item contains a list of chargecodes that can be assigned to sessions or processes run under this usercode if the CHARGEREQ item is set.

- USEDEFAULTCHARGE

  If this item is set, the first chargecode in the user's CHARGECODE list is used as the default chargecode. The default chargecode is used unless the user specifies another chargecode.

If the CHARGEREQ item for a usercode is set, then the Command and Edit (CANDE) MCS or the Menu-Assisted Resource Control (MARC) interface assigns a chargecode to a session at log-on time. If the USEDEFAULTCHARGE item is set for a usercode, then the default chargecode is automatically assigned to the session. Otherwise, the user is prompted to enter a chargecode as part of the log-on procedure. The chargecode is then checked against the CHARGECODE list for the usercode, and rejected if it is not valid.

You can change the chargecode of a CANDE session that is in progress by using the CHARGE command. CANDE verifies that the chargecode is allowed for a usercode before assigning it to the session.

By default, the CHARGE task attribute of a process inherits the CHARGE value of the session or process that initiated it. The value of this task attribute can be specified by task equation or by a task assignment statement. However, in all cases the system software verifies that the new chargecode is allowed for a usercode before assigning it.

The CHARGE task attribute is convenient for grouping all users who belong to a particular department or company. However, CHARGE could be used to indicate a particular type of work instead of a group of people. For example, a usercode can

have two different chargecodes associated with it, each chargecode corresponding to a different work project.

A billing system does not have to use chargecodes. The billing system could charge processes according to their usercodes instead. The billing system could maintain lists of which usercodes belong to which departments and what rates to use when charging each usercode.

Using SMFII for Billing

Before actually running SMFII to generate a billing report, you must create a file containing billing defines. The FILEKIND of the file should be SEQDATA. The defines must be written in the QUERY language, which is defined in the *SMFII Query Operations Guide.*

A list of defines and database items related to log analysis is given in the *SMFII Resource Management Reference Manual.* Many of these log analysis items are useful for billing. The defines listed in the manual are contained in the file DEFINES/SMFII/WORKLOAD/STANDARD, which is supplied with the SMFII software. You can tailor these examples to suit the needs of your site. An example of a define that produces a billing report is also given under "Example 2" in this section.

Once the defines file has been created, you are ready to run SMFII. The first step is to run the LOGCONSOLIDATOR program, which is available to customers in two versions. For billing purposes, the version called SYSTEM/SMFII/LOGCONSOLIDATOR must be used. The other version, SYSTEM/SMFII/LOGCONSOLIDATOR/HWONLY, cannot access the necessary information.

You can run LOGCONSOLIDATOR interactively from CANDE, or initiate it using a Work Flow Language (WFL) job. When initiated by a WFL job, the program searches for input commands in a disk file of type SEQDATA, titled SMFII/LOGCONSOLIDATOR/PARAMETERS. You must either supply such a file or use file equation to cause another file to be used.

The LOGCONSOLIDATOR *ADD* command can be used to generate SMFII database files containing information from one or more system logs. The following is an example of such a command:

```
ADD SUMLOG/503/043086/001074, SYSTEM/SUMLOG FROM DISKB (PACK)
  TO (JASMITH)SMFII/DATA/= ON DPMAST: INCLUDE WORKLOAD(TASKS);
```

If LOGCONSOLIDATOR is being run interactively, you can exit the program with a QUIT command.

The next step is to run the QUERY program. You can run QUERY interactively from CANDE, or initiate it using a WFL job. When initiated by a WFL job, QUERY searches for input commands in a disk file of type SEQDATA, titled SMFII/ANALYSIS/PARAMETERS. You must either supply such a file or use file equation to cause another file to be used.

The first command that should be given to QUERY is one that opens the WORKLOAD database module that was created by the LOGCONSOLIDATOR run. The following is an example of such a command:

```
OPEN WORKLOAD IN (JASMITH)SMFII/DATA ON DPMAST;
```

Next, you must load the defines that will be used. You can either enter them manually, or load them from a file by using the ENTER command. The following is an example of this command:

```
ENTER (JASMITH)TASK/SMFII/DEFINES ON DPFAMILY;
```

Next, you should enter the name of the billing define that you want to be used. For example, if the defines file that you loaded contained a define called SAMPLEBILLING, you could enter

```
SAMPLEBILLING;
```

When QUERY finishes generating the report, you can end the program by using the QUIT command.

The report generated by QUERY is a printer backup file, so if you ran the program from CANDE, you must end the session to generate the printout. If you ran QUERY by using a WFL job, the printout is issued when the job ends.

For more information about SMFII, refer to the *SMFII Resource Management Reference Manual* and the *SMFII Query Operations Guide.*

Example 1

Following is an example of a WFL job that runs LOGCONSOLIDATOR and QUERY to produce a billing report:

```
100 ?BEGIN JOB BILLING/JOB;
110   CLASS = 15;
120   FAMILY DISK = SYSFAM OTHERWISE DISK;
140
150 SUBROUTINE LOGSUM;
160 BEGIN
170   TASK T1;
180
190   RUN SYSTEM/SMFII/LOGCONSOLIDATOR ON SYSFAM;
200     OPTION = (FAULT,ARRAYS);
210     FILE SUMLOG(BLOCKSIZE=15000);
220     FILE SEQDATAFILE(KIND=READER);
230 DATA
240   ADD SUMLOG/= ON DISKB TO (SYS)SMFII/DATA/= ON SYSFAM;
250   INCLUDE WORKLOAD (TASKS);
260 ? % End of LOGCONSOLIDATOR commands
270
280   DO BEGIN
290         T1(STATUS=NEVERUSED);
```

```
300        COPY *SUMLOG/= FROM DISKB(PACK)
310           TO DAILYLOGLF79AB(TAPEPE, SERIALNO="DAILY1")[T1];
320        IF T1 ISNT COMPLETEDOK THEN
330           WAIT("COPY FAILED - OK TO RETRY",OK);
340      END
350   UNTIL (T1 IS COMPLETEDOK);
360
370   REMOVE *SUMLOG/=;
380 END LOGSUM;
390
400 SUBROUTINE BILLSUM;
410 BEGIN
420   RUN SYSTEM/SMFII/QUERY ON SYSFAM;
430     OPTION = (FAULT,ARRAYS);
440     FILE SEQDATAFILE(TITLE=(SYS)TASK/SMFII/DEFINES ON SYSFAM);
450   REMOVE (SYS)SMFII/DATA/= ON SYSFAM;
460 END BILLSUM;
470
475 IF TAKE(TIMEDATE(DDMMYY),2) = "01" THEN
480    BILLSUM;  % ON THE FIRST DAY OF ANY MONTH, MAKE A BILLING REPORT
490 CASE TIMEDATE(MMDDYY) OF
500   BEGIN
510     ("010187", "041787", "052587", "070387", "090787", "112687",
520      "112787", "122487", "122587", "123187"):
530          ; % TODAY IS A HOLIDAY, NO OPERATORS AVAILABLE
540   ELSE:
550          CASE TIMEDATE(DAY) OF
560            BEGIN
570              ("MONDAY","TUESDAY","WEDNESDAY","THURSDAY","FRIDAY"):
580                   LOGSUM;  % RUN LOGCONSOLIDATOR AND REMOVE SUMLOGS
590            END;
600   END;
610 START (SYS)TASK/BILLING/LOGUPDATE ON SYSFAM;
620   STARTTIME = 23:00 ON + 1;
630
640 ?END JOB
```

This job runs LOGCONSOLIDATOR and removes the system log files every weekday that is not a holiday and runs QUERY to produce a billing report on the first of each month.

The FAMILY statement at line 120 is necessary because the SMFII programs are located on a family that is not part of the default FAMILY value for this job.

Subroutine LOGSUM, at lines 150 through 380, updates the SMFII database.

The statement at line 190 runs LOGCONSOLIDATOR. The task equation at line 200 causes a program dump to be taken if the program terminates abnormally. The file equation at line 210 causes LOGCONSOLIDATOR to run more quickly, but also to use more memory than it otherwise would. The file equation at line 220 causes LOGCONSOLIDATOR to read its input commands from the data specification at lines 230 through 260. The ADD command at line 240 causes all released system log files on

DISKB to be analyzed and the logging information to be added to the WORKLOAD module under usercode (SYS) on SYSFAM family. The WORKLOAD module consists of the following two files:

```
SMFII/DATA/WORKLOAD/TASKHISTORY
SMFII/DATA/WORKLOAD/SYMBOLTABLE
```

The statements at lines 280 through 350 copy the released system log files to tape. The copy operation will wait on a NO FILE condition until the system operator loads a tape. When the copy operation has succeeded, the statement at line 370 removes the released system log files from disk. (The system log file currently in use is not affected.)

Subroutine BILLSUM, at lines 400 through 460, produces a billing report. The statement at line 420 runs QUERY. The file equation at line 440 causes QUERY to read its input commands from the file (SYS)TASK/SMFII/DEFINES ON SYSFAM.

The statement at lines 475 through 480 calls the subroutine BILLSUM to produce a billing report on the first day of each month.

The CASE statement at lines 490 to 600 calls the subroutine LOGSUM on selected days. LOGSUM is not run on weekends or holidays because an operator must be present to load a tape when LOGSUM is run.

The START statement at lines 610 and 620 causes the job to be run again the next day at 11:00 p.m.

Example 2

The following example shows the contents of the file TASK/SMFII/DEFINES. These commands are sufficient to produce a simple billing report summarizing charges for each chargecode on the system:

```
100 OPEN WORKLOAD IN (SYS)SMFII/DATA ON SYSFAM;
110
120 DEFINE
130        SAMPLEBILLING =
140            TAB LOGCHARGECODE
150               ,PROCTIME
160               ,IOTIME
170               ,LINESPRINTED
180               ,CHARGEDOLLARS
190            SORT BY ITEM 1
200            :TABTITLE = NEWLINE(1) & "SAMPLE BILLING REPORT"
210            ,TABFORMAT = (AL011,X6,AR08,X6,AR08,X6,I9,X8,AR07)
220            ,DEST = PRINTER, PAGEWIDTH=72, PAGELENGTH = 50 #
230       ,PROCTIME = SECONDSTOHHMMSS(SUM(LOGPROCESSORTIME)) #
240       ,IOTIME = SECONDSTOHHMMSS(SUM(LOGIOTIME)) #
250       ,LINESPRINTED = (SUM(LOGPRTINFO)) #
260       ,CHARGEDOLLARS = CHARGES(SUM(LOGPROCESSORTIME)
270                                ,SUM(LOGIOTIME)
280                                ,MEAN(LOGPRIORITY)
290                                ,SUM(LOGPRTINFO)) #
```

```
300       ;
310
320 EXPAND
330        LOGCHARGECODE = "CHARGE CODE"
340       ,CHARGEDOLLARS = "DOLLARS"
350       ;
360
370 STRING PROCEDURE CHARGES(PROCTIMEV
380                          ,IOTIMEV
390                          ,PRIORITYV
400                          ,LINESPRTDV
410                          )[MAXSIZE=16];
420 REAL
430     PROCTIMEV,
440     IOTIMEV,
450     PRIORITYV,
460     LINESPRTDV;
470 BEGIN
480 STRING
490        DOLLARSTR
500       ,CENTSTR
510       ;
520 REAL
530      PENNIES
540     ,DOLLARS
550     ,CENTS
560     ;
570
580 PENNIES := (((PROCTIMEV / 0.6) + (IOTIMEV / 0.6))
590           * (PRIORITYV / 50)) + (LINESPRTDV / 5);
600
610 DOLLARS :=  PENNIES DIV 100;
620 DOLLARSTR := PREFIX(EBCDIC(DOLLARS,*),".");
630 CENTS := ((PENNIES DIV 1) - (DOLLARS * 100));
640 CENTSTR := TAKE(EBCDIC(CENTS,*) & "0",2);
650
660 CHARGES := DOLLARSTR & "." & CENTSTR;
670
680 END;
690
700 STRING FUNCTION SECONDSTOHHMMSS(ITEM)[MAXSIZE=8];
710 REAL
720     ITEM;
730 BEGIN
740 STRING
750     S
760     ;
770 REAL
780     HH
790     ,MM
800     ,SS
810     ;
```

```
820 HH :=  ITEM DIV 3600;
830 MM := (ITEM MOD 3600) DIV 60;
840 SS := (ITEM MOD 60);
850
860 IF HH GEQ 10 THEN
870     S := S & EBCDIC(HH,2)
880 ELSE
890     S := S & " " & EBCDIC(HH,1);
900 S := S & ":"
910        & EBCDIC(MM,2)
920        & ":"
930        & EBCDIC(SS,2)
940      ;
950 RETURN(S);
960 END;
970
980 SAMPLEBILLING;
```

The overall result of this example is to create a billing report based on chargecodes. Each chargecode is listed in the billing report, followed by the total processor time, total I/O time, total lines printed, and total billing charge for that chargecode.

In this example, the statement at line 100 opens the database WORKLOAD module. The statements at lines 120 through 960 create a billing define called SAMPLEBILLING, together with various other defines and procedures that SAMPLEBILLING calls to do its work. The statement at line 980 causes the execution of the SAMPLEBILLING define and thus produces a billing report.

The DEFINE command at line 120 creates several defines, of which SAMPLEBILLING is the first. SAMPLEBILLING is defined as a call on the TABULATE command, which is described in detail in the *SMFII Query Operations Guide*. The TABULATE command used here constructs a table with five columns of information. The first column contains chargecodes, which are obtained from the database item LOGCHARGECODE. The following four columns list information from the four defines PROCTIME, IOTIME, LINESPRINTED, and CHARGEDOLLARS. The meaning of each of these is specified at lines 230 through 290.

The EXPAND command at line 320 specifies spelled-out versions for two of the column headings.

The procedure CHARGES, at lines 370 through 680, does the actual work of computing the charge. The algorithm that computes the charge is at lines 580 and 590. You can create a different algorithm to reflect the billing needs of your site.

The algorithm shown is based on four factors: total processor time, total I/O time, average priority, and number of lines printed. A dollar is charged for each minute of processor time and for each minute of I/O time. The charge is then adjusted to reflect the average priority, with a higher priority being charged more. Additionally, one dollar is charged for each 500 lines printed.

The procedure SECONDSTOHHMMSS, at lines 700 through 960, renders the processor time and I/O time values into a neat format.

An example follows of the first few lines of output from the SAMPLEBILLING define:

```
                    SMFII ANALYSIS FOR SYSTEM #503
              TIME RANGE: 00:02:18 10/30/90 - 15:21:01 10/30/90

                          SAMPLE BILLING REPORT
```

CHARGE CODE	PROCESSOR TIME	IOTIME	LINES PRINTED	DOLLARS
.	1:56:09	11:13:07	37833	1161.11
7251	0:02:13	0:04:16	10044	26.26
7252	0:10:35	0:08:44	0	19.19
7252	0:00:01	0:00:00	0	0.10
7421	0:00:01	0:00:02	0	0.50
7462	0:00:07	0:00:13	0	0.33
7463	0:00:13	0:00:21	4801	10.10
7622	0:22:11	0:21:23	7863	60.60

Using LOGGER for Billing

You can run the LOGGER utility to produce a summary of resource usage for all processes having a given chargecode or a given usercode. If you have created and installed a BILLINGSUPPORT library, you can also use LOGGER to produce a summary of charges for each charge code or usercode. For further information about this library, refer to "BILLINGSUPPORT Library" later in this section.

LOGGER operates in two main phases. During the first phase, it reads a system log file and creates data files containing the information necessary to generate a report. During the second phase, LOGGER reads the data files and generates a report.

The data files created during the first phase are the JOBSUMMARY file, the STATISTICS file, and, optionally, a FILEIODATA file and a DRCDATA file. Of these, the JOBSUMMARY file contains the information most likely to be of use for billing: data about each job, task, and session.

You can use LOGGER to generate reports covering extended time periods such as a week, a month, or a year. A year-to-date totals file, called the YTDFILE, plays an important role in the creation of such reports. The LOGGER OPTION UPDATE command causes a YTDFILE to be created or updated. The LOGGER OPTION YEAR command causes a year-to-date report to be generated from the YTDFILE. By removing the YTDFILE periodically, you can ensure that the current YTDFILE contains a summary of information for the current week or the current month only, rather than for the entire year to date.

LOGGER expects to find its input commands in a card reader file titled CARD. A simple file equation can be used to make LOGGER read commands from a disk file instead. The

following command, if entered during a CANDE session, would run LOGGER and cause LOGGER to read commands from a disk file titled BILLING/INPUT:

```
RUN *SYSTEM/LOGGER ON DISK;FILE CARD(KIND=DISK,TITLE=BILLING/INPUT);
```

If initiated by a WFL job, LOGGER can be made to read its input from a data specification in the WFL job. An example of such a WFL job is given under "Example 3" in this section.

The report generated by LOGGER is a printer backup file, so if you ran the program from CANDE, you must end the session to generate the printout. If you ran LOGGER by using a WFL job, the report is automatically queued for printing when the WFL job terminates.

For detailed information about LOGGER, refer to the *System Software Support Reference Manual.*

Example 3

The following sample WFL job runs LOGGER every day and produces both a daily and a monthly billing report. The reports that are produced are similar to that generated by the sample SMFII billing job earlier in this section.

```
110 ?BEGIN JOB TASK/BILLING;
120    CLASS = 15;
130    JOBSUMMARY = SUPPRESSED;
140    FAMILY DISK = SYSFAM OTHERWISE DISK;
150
160 IF (TAKE(TIMEDATE(DDMMYY),2) = "Ø1")
170     AND FILE YTDFILE IS RESIDENT THEN
180 BEGIN
190    RUN *SYSTEM/LOGGER ON DISK;
200 DATA
210 OPTION YEAR
220 ?
230    REMOVE YTDFILE;
240 END;
250 DISPLAY ("TL THE CURRENT LOG AND ENTER " &
260            STRING(MYSELF(MIXNUMBER),*) & " HI COMMAND");
270 WAIT;
280 RUN *SYSTEM/LOGGER ON DISK;
290 DATA
300 USE SUMLOG CURRENT
310 OPTION UPDATE
320 REPORT
330 SOURCE FILE IS JOBSUMMARY
340 SORT BY CHARGECODE ASCENDING
350 BREAK ON CHARGECODE TOTALING PROCESSTIME, IOTIME, LINES, CHARGES
360 OUTPUT ITEMS ARE PROCESSTIME, IOTIME, LINES, CHARGES
370 REPORTS ARE SUMMARY 1
380 HEADING IS "LOGGER BILLING EXAMPLE"
390 END
400 STOP
410 ?
420 REMOVE JOBSUMMARY/=, STATISTICS/=;
430 START (JASMITH)TASK/BILLING ON SYSFAM;STARTTIME=23:ØØ ON + 1;
440 ?END JOB
```

The FAMILY statement at line 140 causes SYSFAM to be used as the default family for any input or output files used by the job. Thus, the output files JOBSUMMARY, STATISTICS, and YTDFILE are created on SYSFAM.

The statement at line 160 checks to see whether the current day is the first of the month. If so, the statement at line 170 checks to see whether a YTDFILE summarizing data for the last month is available. If so, the statement at lines 190 to 220 runs LOGGER and causes it to read the YTDFILE and produce a monthly billing report. After the report is produced, the statement at line 230 removes the YTDFILE.

The statement at lines 250 and 260 asks the system operator to use the TL (Transfer Log) system command to save the current system log and create a new system log. The saved system log is automatically retitled according to the convention described under "System Log File" in Section 11. The statement at line 270 causes the job to wait until the operator enters a HI (Cause Exceptionevent) system command to signify that the TL operation is complete.

The statement at line 280 initiates the LOGGER utility. LOGGER reads the commands from the data specification at lines 290 to 410. The statement at line 300 causes LOGGER to read the system log files that have today's date in the title. The statement at line 310 causes LOGGER to update a year-to-date file titled YTDFILE. The statements at lines 320 to 390 cause LOGGER to produce a report showing the processor usage, I/O usage, lines printed, and a billing charge for each charge code. (Note that the billing charge will be zero in each case, unless a BILLINGSUPPORT library is installed.) The statement at line 420 then removes the output files.

The statement at line 430 initiates a new instance of the job, with a STARTTIME of 11:00 p.m. the following day.

An example follows of the first few lines of output from this job:

```
                    LOGGER BILLING EXAMPLE
                         11/29/90

                SUMMARY REPORT BY CHARGECODE
```

CHARGE CODE	TOTAL PROCESSOR TIME	TOTAL IOTIME	LINES	TOTAL CHARGES	TOTAL NUMBER OF RUNS
	2887.00	31039.73	2764	$49446.30	891
7251.	9.50	10.72	0	$49.44	6
7252.	47.19	93.68	2140	$410.46	36
7261.	32.09	24.03	0	$87.51	4
7462.	9.50	19.10	0	$157.70	10
7463.	40.60	12.98	0	$280.91	28
7500.	3.46	0.11	0	$8.45	1
7501.	0.23	0.00	0	$47.10	1
7622.	981.00	772.98	13288	$7197.48	36

Real-Time Billing Options

The real-time billing system permits selective invocation of billing library entry points. These billing options must be initialized in the billing library before initiating the billing system. You can select any combination of billing options. The options currently available to real-time billing are

- JOBFORMATTER processing
- Visible task beginning and end
- Session beginning and end
- Status enquiries for MARC and CANDE

All invocations of the billing library by system software are made from the operating system. CANDE and MARC refer to the selected billing options after linking the MCPSUPPORT library. Billing options cannot be changed unless the billing library is terminated and reinitialized with the billing options word set to the new value. The operating system only links to the billing library during system initialization. A halt/load is required for the operating system to recognize new billing options, or to change the BILLINGSUPPORT library.

The real-time billing library has one exported entry point to manage billing selections. This entry point is used by the operating system, JOBFORMATTER, CANDE and MARC to determine which billing options have been set. A complete description of this entry point is provided in "BILLINGSUPPORT Library" in this section.

BILLINGSUPPORT Library

BILLINGSUPPORT is the name of the library that is called by the operating system and the LOGGER utility to compute billing charges. Unisys does not supply this library as such, but provides a sample version of the library explaining the billing options word, the entry points, and the required parameters. This sample version of the library is called SYMBOL/BILLINGSUPPORT.

It is not necessary for a site to have a BILLINGSUPPORT library. However, you might want to create one for either of the following reasons:

- To make it possible to generate billing reports through the LOGGER utility. When LOGGER is used to return the CHARGES data item in the JOBSUMMARY file, the DRCDATA file, or the FILEIODATA file, LOGGER calls the BILLINGSUPPORT library to compute the charge. If no BILLINGSUPPORT library is available, a charge of zero dollars and zero cents is returned.
- To implement a real-time billing system. This system provides billing for CANDE and MARC users, for all visible tasks, and for JOBFORMATTER. (JOBFORMATTER processes job summary information upon the termination of each job.) CANDE, MARC, JOBFORMATTER, and the operating system use BILLINGSUPPORT to generate billing information and to return messages for display to CANDE and MARC users, or to JOBFORMATTER for inclusion in the printed job summaries.

BILLINGSUPPORT receives real-time billing information reliably from all system software, with the possible exception of the JOBFORMATTER interface. Any programmer can prevent JOBFORMATTER from being run for a job by setting the JOBSUMMARY task attribute to SUPPRESSED. A programmer can also prevent any information from being stored in the job log by setting the NOJOBSUMMARYIO task attribute to TRUE. In either of these cases, a real-time billing system cannot compute any charges for that job or any of its tasks by using the JOBFORMATTER interface.

The BILLINGSUPPORT library that you create can have any title. The following system command causes the library to be recognized as the BILLINGSUPPORT library for the system:

```
SL BILLINGSUPPORT = <code file title>
```

The JOBFORMATTER source program includes log analysis defines that can be used in a BILLINGSUPPORT library. You can include these defines in the BILLINGSUPPORT library through the use of the $INCLUDE compiler control option. For an example of such an INCLUDE, refer to the sample BILLINGSUPPORT library source program later in this section.

The following paragraphs outline the features that must be included in a BILLINGSUPPORT library in order for LOGGER, CANDE, MARC, the operating system and JOBFORMATTER to receive bill processing.

- BILLINGSUPPORT should be a SHAREDBYALL library that is frozen permanently. If it is not, care must be exercised to prevent a loop caused by the termination of one BILLINGSUPPORT invocation calling JOBFORMATTER to print its job summary, which causes another BILLINGSUPPORT library to be initiated, which then terminates, causing another JOBFORMATTER invocation and so on.
- The library objects exported by the BILLINGSUPPORT library must be called LOGRECORDBILL, MCP_TASKBILLER, MCS_TASKBILLER, and RETRIEVE_BILLING_OPTIONS. Only LOGRECORDBILL is used for historical billing; real-time billing requires all library objects. LOGRECORDBILL, MCP_TASKBILLER, and MCS_TASKBILLER must return an integer value which is interpreted as the charge in billing units. (Note that LOGGER assumes this integer value to be expressed in cents.) Returning a negative value can be used to indicate an error.

If a situation arises where the BILLINGSUPPORT library has become corrupted, or if you need to remove your current BILLINGSUPPORT library after it has been linked to your operating system through the *SL BILLINGSUPPORT* system command, you must perform the following steps:

1. Remove the object code file for the current BILLINGSUPPORT library.
2. Reinitialize the operating system by performing a halt/load.

   ```
   ??PHL
   ```

3. Remove the link to the existing BILLINGSUPPORT file by entering the following system command:

   ```
   SL- BILLINGSUPPORT
   ```

4. Re-create your BILLINGSUPPORT library. Unisys provides a sample library file called SYMBOL/BILLINGSUPPORT. The file components, entry points and required parameters are all explained in this file.
5. Initiate the new BILLINGSUPPORT library by entering the following system command with the name of the new BILLINGSUPPORT library file:

   ```
   SL BILLINGSUPPORT = <code file title>
   ```

6. Initiate your operating system again by performing another halt/load:

   ```
   ??PHL
   ```

After reinitialization completes, your system should be linked to the new BILLINGSUPPORT library file.

LOGRECORDBILL Parameters

The LOGRECORDBILL procedure is used for historical and real-time billing. The LOGGER utility uses this procedure for historical billing. JOBFORMATTER, CANDE, and MARC use LOGRECORDBILL for real-time billing. JOBFORMATTER uses this procedure to process billing information for job summaries. CANDE and MARC use LOGRECORDBILL for billing, bill processing for the beginning and ending of sessions, and for status inquiries. The LOGRECORDBILL requires the following four parameters:

- Caller Identification
- Resource Usage
- Lines to Print
- Work Area

These parameters are described in the following information.

Caller Identification

The Caller Identification parameter is of type real and is passed by value. This parameter allows the LOGRECORDBILL procedure to identify the nature of the processing being requested from it. It has the following format:

Field	Meaning
[07:08]	1 = JOBFORMATTER
	2 = LOGGER
	3 = CANDE (beginning of session)
	4 = CANDE (end of session)
	5 = CANDE (session status)
	6 = MARC (beginning of session)
	7 = MARC (end of session)
	8 = MARC (session status)
	9-16 = Values used by other BILLINGSUPPORT entry points.
	17 = FILEIODATA report
	18 = DRCDATA report
	19-127 = Values reserved for future system software use.
	> 127 = The caller is a user program.

continued

continued

Field	Meaning
[47:40]	This field is available for user definition when the calling program is not system software. System software might use this field at a future date.

There is no requirement that each caller have a unique identification. There is also nothing to forbid a user program that wishes processing identical to that provided for LOGGER from setting [07:08] = 2.

Resource Usage

The Resource Usage parameter is a real array of variable length that is passed by reference. System software passes an array in the format of a log record. The formats vary across the different LOGRECORDBILL callers. Listed below is a description of the possible log records passed from system software. You can find details of the log record format in the information on SUMLOG in the *System Software Support Reference Manual.*

Caller	Log Format	Major Type	Minor Type
JOBFORMATTER	Various formats		
LOGGER	EOJ entry	1	2
LOGGER	EOT entry	1	4
LOGGER	MCS log off	4	2
CANDE (beginning of session)	MCS log on	4	1
CANDE (end of session)	MCS log off	4	2
CANDE (session status)	none (see below)	-	-
MARC (beginning of session	MCS log on	4	1
MARC (end of session)	MCS log off	4	2
MARC (session status)	none (see below)	-	-

For the two session status cases, a partial MCS log off record is passed in which the following fields are valid:

```
Word  Ø [15:16] - session number
Word 11         - processor time
Word 12         - I/O time
Word 23         - session elapsed time
```

LOGRECORDBILL should not change the contents of this array, because the calling program might not be finished using the array.

Lines to Print

The Lines-to-Print parameter is a real array that is passed by reference. The system software does not use this parameter to pass any information to the procedure. Rather, it is provided so that LOGRECORDBILL can pass back information for use by the calling programs. CANDE and MARC display this information to the user. JOBFORMATTER includes this information in job summaries. JOBFORMATTER truncates any print lines longer than 120 characters.

The first byte of the array indicates, in binary, the number of lines to be printed. The remainder of the array contains the print lines, each terminated by a null character.

It is the responsibility of LOGRECORDBILL to ensure that the array is large enough to contain all the data it wishes to return, and to resize the array if necessary. If the calling program is system software, it provides the array with a minimum size of one word.

Callers doing printing should initialize word 0 to zero before each call. The following is a simple schematic program fragment illustrating the method the caller uses to print the lines:

```
POINT := POINTER(PARM3ARRAY);
THRU REAL(POINT,1) DO
  BEGIN
    REPLACE PRINTBUFFER BY POINT:POINT+1
      UNTIL = 48"00";
    WRITETOPRINTER(PRINTBUFFER);
  END;
```

Work Area

The Work Area parameter is a real array that is passed by reference. This array is provided by the calling program for the LOGRECORDBILL procedure to use as a work area. The intent is to provide a mechanism for accumulating charges from several resource usage records.

It is the responsibility of LOGRECORDBILL to define the format of this array and, if necessary, to resize it to the proper size.

The calling program should not initialize this array or change its content in any way. The calling program can choose to provide a new array for each new job number it processes and to pass that same array for all log records pertaining to the same job. LOGRECORDBILL can use this array to accumulate charges for a job. JOBFORMATTER provides a new array for each job, but LOGGER, CANDE, MARC, and the operating system do not.

MCP_TASKBILLER Parameters

MCP_TASKBILLER is invoked by the operating system when bill processing is required at the beginning or end (or both) of jobs and visible tasks. MCP_TASKBILLER requires the following three parameters:

- Caller identification
- Resource usage
- Task information

These parameters are described in the following information.

Caller Identification

This identifier has the same function as for LOGRECORDBILL and is used by MCP_TASKBILLER to determine the nature of the processing requested. It is of type real and is passed by value. The definition of caller is expanded as follows:

Field	Meaning
[07:08]	9 = MCP (beginning of job or task) 10 = MCP (end of job or task) > 127 = The caller is a user program
[47:40]	This field is available for user definition when the calling program is not system software.

Resource Usage

This parameter is identical to what is defined for LOGRECORDBILL. It is a real array of variable length passed by reference. System software sends BILLINGSUPPORT a log record in this array for bill processing. The log record formats for each call to MCP_TASKBILLER are given below. You can find details of the log record formats in the information on SUMLOG in the *System Software Support Reference Manual.*

Caller	Log Format	Major Type	Minor Type
MCP (beginning of job)	BOJ entry	1	1
MCP (beginning of task)	BOT entry	1	3
MCP (end of job)	EOJ entry	1	2
MCP (end of task)	EOT entry	1	4

Task Information

This parameter is a task variable passed by reference. System software uses this parameter to give BILLINGSUPPORT access to the attributes of the current task.

MCS_TASKBILLER Parameters

MCS_TASKBILLER provides bill processing for tasks initiated from an MCS. The real-time billing system invokes MCS_TASKBILLER for CANDE and MARC at the beginning of a task, the ending of a task, or when status inquiries are made on a task. This procedure provides bill processing for tasks initiated through the MCS and may return billing reports for display to users. The three following parameters are required:

- Caller Identification
- Lines to Print
- Task Information

These parameters are described in the following information.

Caller Identification

This parameter has the same function as for LOGRECORDBILL. This identifier is of type real and is passed by value. It is used by MCS_TASKBILLER to determine the nature of the processing requested. The definition of CALLER is expended as follows:

Field	Meaning
[07:08]	11 = CANDE (beginning of task) 12 = CANDE (end of task) 13 = CANDE (status enquiry) 14 = MARC (beginning of task) 15 = MARC (end of task) 16 = MARC (status enquiry) > 127 = The caller is a user program
[47:40]	This field is available for user definition when the calling program is not system software.

Lines to Print

This parameter is a real array that is passed by reference. The billing library uses this array to pass billing messages for display by the calling program. This parameter is identical to what is required by LOGRECORDBILL. Refer to "LOGRECORDBILL Parameters" in this section for more information.

Task Information

This parameter is a task variable passed by reference. System software uses this parameter to give BILLINGSUPPORT access to the task attributes of the current task.

RETRIEVE_BILLING_OPTIONS Parameters

This entry point is invoked by CANDE, MARC and JOBFORMATTER when the status of the billing selections is required. CANDE and MARC access this entry point while initializing.

RETRIEVE_BILLING_OPTIONS does not have any parameters. This procedure returns a real value indicating the selected billing options.

BILLING_OPTIONS is the real value word in the billing library which stores the options of the billing system. The state of the bit corresponding to the CALLER identification value reflects the setting and resetting of the billing option.

Calling Programs

Currently, the BILLINGSUPPORT library is used by JOBFORMATTER, the operating system, CANDE, MARC, and LOGGER.

JOBFORMATTER

JOBFORMATTER calls the LOGRECORDBILL procedure for each record read from the job log when the JOBFORMATTER billing option is set and JOBFORMATTER has completed processing for that record.

After the call, JOBFORMATTER writes any print lines returned in the Lines-to-Print parameter. This parameter enables LOGRECORDBILL to cause the total charge to be displayed in the job summary.

JOBFORMATTER provides a new array for the Work Area parameter each time it processes a job summary. However, the same array is used when processing each record of a given job summary. LOGRECORDBILL can use this array to accumulate the total charges for the job.

JOBFORMATTER ignores the value returned by LOGRECORDBILL, even if the value is negative, indicating an error.

The system runs JOBFORMATTER once at the end of each job, unless the job summary for that job is suppressed. If the system halt/loads while a job summary is being printed, then a new JOBFORMATTER run reprocesses the entire job log. Therefore, it is possible that JOBFORMATTER will run more than once for a particular job or session.

Note that while all the log records in the job log are passed to LOGRECORDBILL, the job log might not include all the possible types of log records. You can use selective logging features to specify which types of log entries are written to job logs. For an overview of selective logging features, refer to the *System Software Support Reference Manual*. The job log can also be incomplete if job log overflow has occurred.

To provide additional information, JOBFORMATTER defines the following fields in the first parameter:

Field	Meaning	
[08:01]	1	This is the first call on LOGRECORDBILL by this run of JOBFORMATTER. There is a valid resource usage record in the second parameter, but it is not guaranteed to be a BOJ entry or BOT entry.
[09:01]	1	This is the last time this run of JOBFORMATTER will call LOGRECORDBILL, because processing of this job summary has been completed. No resource record exists in the second parameter. The only reason for this call is to allow LOGRECORDBILL to wrap up its processing for this job.

continued

continued

Field	Meaning	
[10:01]	1	The job log is incomplete because of job log overflow or because an I/O error occurred while JOBFORMATTER was reading the job log. This field is valid only when [09:01] = 1.

Operating System

The operating system requires the MCP_TASKBILLER procedure to perform billing for all visible tasks. The billing library is invoked just after making BOT and EOT log entries. Billing is not provided at BOT for library tasks.

The MCP ignores all values returned from MCP_TASKBILLER.

CANDE

CANDE requires LOGRECORDBILL and MCS_TASKBILLER to provide billing support. CANDE may use BILLINGSUPPORT at the beginning as well as at the end of sessions or tasks. Billing functions may also be performed when status enquiries are requested. LOGRECORDBILL is called when session information is to be billed. This includes processing status enquiries when a session is dormant or performing CANDE commands (for example, FIND MATCH, REPLACE). MCS_TASKBILLER is used to process billing for tasks initiated from the MCS session. MCS_TASKBILLER processes status enquiries when such tasks are running.

Before each call to the billing library, CANDE initializes the first byte of the print lines array to zero. BILLINGSUPPORT returns the print lines array to CANDE at the completion of bill processing in the Lines to Print parameter.

CANDE does not use the Work Area parameter of LOGRECORDBILL, and CANDE ignores the values returned from LOGRECORDBILL and MCS_TASKBILLER.

MARC

The real-time billing capabilities of MARC are very similar to those of CANDE. Like CANDE, MARC uses LOGRECORDBILL to process session information and MCS_TASKBILLER to process task information.

Bill processing is provided at the beginning and end of sessions and tasks as well as for status inquiries. The WRU selection from the SC menu produces a screen displaying any session status information and billing messages returned from BILLINGSUPPORT. To display billing messages when the program uses a remote file, use the Y menu selection available on the TASKSTATUS menu.

MARC can request that bill processing be performed whenever any of the events listed in Table 12–1 occur. MARC displays billing messages whenever some, but not all, of the listed events occur.

Table 12–1. Sample Billing Utility Bill Processing Schedule

Event	Billing Messages Displayed
Beginning of session (BOS)	None
End of session (EOS) initiated by HELLO command	All
End of session (EOS) initiated by SPLIT command	All
End of session (EOS) initiated by BYE command	None
End of session (EOS) initiated by COMS command CLOSE	None
Beginning of task (BOT)	All
End of task (EOT)	All
Session or task status inquiry	All

Billing messages for the beginning or end of sessions are displayed on the bottom two lines of the home screen. If the message is longer than two lines, a separate screen is provided.

Billing information for tasks is displayed on the task status screen.

The billing library returns all billing messages in the Lines to Print parameter.

MARC ignores the values returned from LOGRECORDBILL and MCS_TASKBILLER.

LOGGER Utility

When LOGGER calls the LOGRECORDBILL procedure, the value returned is placed in the JOBSUMMARY, FILEIODATA, or DRCDATA file item CHARGES. If an error is returned, LOGGER stores a zero, but takes no further action. When the value is stored, LOGGER assumes that the billing unit is cents and places the decimal point(.) and dollar sign ($) accordingly.

LOGGER calls the LOGRECORDBILL procedure only when processing the end of job, end of session, and end of task log records. Therefore, the charge returned to LOGGER by LOGRECORDBILL must be based on information available in those types of log records.

LOGGER has not finished its processing of the Resource Usage array when the LOGRECORDBILL procedure is called. Therefore, LOGRECORDBILL must not change the value of the Resource Usage parameter. LOGGER does not use the Lines-to-Print parameter. Also, LOGGER does not provide a new work array for the Work Area parameter at any point.

Sample BILLINGSUPPORT Library

The following is a simple example of a BILLINGSUPPORT library source program. This example is intended to be used only with LOGGER; it does not compute a charge if called by JOBFORMATTER, CANDE, or MARC.

```
100  $ SHARING = SHAREDBYALL
110 BEGIN
120
130  $ INCLUDE NEWJOBFORMATTER 300000 - 710000
140  $ INCLUDE NEWJOBFORMATTER 10000000 - 15010000
150    DEFINE RECORD            = BUF#,
160           PRICECPUTIME      = 37#,
170           PRICEIOTIME       = 47#,
180           PRICEMEMINT       = 0.8#,
190           PRICEPRINTLINES   = 1.7#,
200           PRICECARDSREAD    = 3#,
210           PRICECARDSPUNCHED = 3#,
220           PRICEPERTASK      = 700#,
230           PRICECANDETIME    = 2000/3600#;
240
250 INTEGER PROCEDURE LOGRECORDBILL (CALLER
260                                 ,RECORD
270                                 ,PRINTLINES
280                                 ,WORKAREA);
290   VALUE CALLER;
300   REAL CALLER;
310   REAL ARRAY RECORD, PRINTLINES, WORKAREA [0];
320
330 BEGIN % OUTER BLOCK OF LOGRECORDBILL PROCEDURE
340    IF CALLER.[7:8] = 2   % IF THE CALLER IS LOGGER
350    THEN
360        BEGIN
370            IF TYPEW.MAJORF = LOGMAJMCS
380            THEN    % CANDE SESSION
390                LOGRECORDBILL := PRICECANDETIME * ELAPSEDTIME
400                                 * 2.4/1000000
410            ELSE
420            IF TYPEW.MAJORF = LOGMAJJOB
430            THEN    % EOJ/EOT
440                LOGRECORDBILL := PRICECPUTIME * PROCTIME
450                                 * 2.4/1000000
460                               + PRICEIOTIME * IOTIME
470                                 * 2.4/1000000
480                               + PRICEMEMINT * (SVMEM + OVMEM)
490                                 * 2.4/1000000000
500                               + PRICEPRINTLINES * LINES
510                               + PRICECARDSREAD * CRD
520                               + PRICECARDSPUNCHED * CRDPCH
530                               + PRICEPERTASK;
540        END;
550 END OF LOGRECORDBILL;
560
570 EXPORT LOGRECORDBILL;
580
590 FREEZE (PERMANENT);
600
610 END PROGRAM.
```

The statement at line 100 specifies that this is to be a SHAREDBYALL library. The statement at line 590 specifies that the library is to be frozen permanently.

The INCLUDE options on lines 130 and 140 cause defines from the JOBFORMATTER source program to be included in the program. The JOBFORMATTER defines are used to analyze log entries.

The defines in the JOBFORMATTER source program extract information from an array called BUF. The define at line 150 identifies BUF as the array passed to the RECORD parameter of the LOGRECORDBILL procedure.

The defines at lines 160 through 230 specify the amounts of money to be charged for various types of resource usage.

The LOGRECORDBILL procedure extends from lines 330 to 560. Line 340 checks the CALLER parameter; no further action is taken unless the caller is LOGGER. Additionally, the statements at lines 370 and 420 inspect the major type of the log entry. A charge is computed only for MCS Entries (Major Type 4) and Job and Task Entries (Major Type 1).

The statement on lines 370 to 400 computes a charge for sessions. ELAPSEDTIME is a JOBFORMATTER define that extracts the session elapsed time from the Log-off Entry (Major Type 4, Minor Type 2). The elapsed time value is multiplied by the PRICECANDETIME value defined at line 230.

The statement at lines 410 to 530 computes a charge for jobs and tasks. PROCTIME, IOTIME, SVMEM, OVMEM, LINES, CRD, and CRDPCH are all JOBFORMATTER defines that extract resource usage information from the EOT Entry (Major Type 1, Minor Type 4). The usage information is multiplied by the charge rates that were defined at lines 160 through 230.

This sample BILLINGSUPPORT library can be used by LOGGER to generate a billing report such as the one shown under "Example 3" earlier in this section.

Section 13
Analyzing and Reporting System Problems

This section addresses how to approach a problem and how to report a problem to Unisys or make a suggestion to Unisys.

Analyzing Your Problem

Communicating information about problems can be a difficult task. Knowing how to define a problem can ease the process and enable Unisys to help you quickly and efficiently.

Ideally, to define the problem, you isolate, reproduce, and identify it. If this is not possible, your goal becomes the identification of the essential characteristics of the problem. In this case, it is important that your definition be as specific as possible.

While it is tempting to use a random approach in defining a problem, experience has shown that a structured approach works best. A structured approach can be divided into three phrases:

- Make quick checks.
- Ask basic questions.
- Break the problem down.

Phase 1: Make Quick Checks

The purpose of the quick checks is to determine if the problem is being caused by hardware or by software. This is necessary because something that appears to be a software problem can in fact be a hardware or operational problem.

Take care to make all the following checks. Some of them are easy to overlook if you are pressured for time.

Quick Hardware Checks

Find the answers to the following questions:

- Have there been any hardware changes or additions recently?
- Are all interface cables installed and secure?
- Do any of the devices involved report hardware errors?

Quick Operational Checks

Determine the answers to the following questions:

- Do all units show power on and ready status?
- Are all peripheral switches in the proper positions for the job or applications?
- Is all necessary software installed?
- Are all configuration files resident and do they contain correct values?
- Are all required files resident?
- Have any of the files involved recently been re-created or reloaded?
- Have any of the devices involved recently required maintenance?
- Are file, path, volume, and family names correct?
- Are there sufficient resources to run the job?
- Is the security level correct?
- Have operational procedures changed recently?
- Have new operators been using the system?

Checking these items often solves the problem. However, if this check does not reveal any causes, proceed to phase 2.

Phase 2: Ask Basic Questions

At this point, you already should have ruled out hardware and operational problems as the cause of the problem. Phase two involves asking basic questions about what has changed. This can lead directly to the cause of a problem, because problems often appear when change occurs. Ask the following basic questions about change:

- Has a new maintenance release or patch been installed recently?
- Were unique types of data being run at the time the problem occurred?
- Did the volume of data input change prior to the occurrence of the problem?
- Has anyone recently modified any of the programs that were being run at the time the problem occurred?
- Have there been any environmental software changes recently?
- Have you recently identified a problem in another software product?
- Were new functions or features being used at the time the problem occurred?
- Did input parameters or control files change prior to the occurrence of the problem?

If your answer to any of these questions is yes, the problem might be related to the change the question identified. If you can, test the suspected cause by reversing the change. If you feel confident that you have found the problem, you are ready to report it.

At this point, try to reproduce the problem. If you can reproduce it, try to develop a test case or procedure so that Unisys service people can also reproduce it. However, if you still cannot define the problem, proceed to phase 3.

Phase 3: Break the Problem Down

Phase 3 is more complex than phases 1 and 2, because the probable causes are more elusive. To find these causes, you break the problem down progressively. First, you identify the essential conditions of the problem, then you break those conditions down into their conditions, and so on, until you can define the problem.

Identifying the Essential Conditions

You identify the essential conditions by stating as much as possible about exactly what happens when the problem occurs. To do this, find answers to the following questions:

- When does the problem occur?
- What programs were running when the problem occurred?
- What did the operator do just before the problem occurred?
- How many users were online and what was each doing just before the problem occurred?

Usually after you have asked these questions, you find that you have many conditions to work with. If you do not have enough information, try to duplicate the problem.

Now try to narrow down the list of essential conditions by testing the conditions in the list. Try various combinations of conditions until you are satisfied that you have all, and only, the essential conditions. Start by asking the following questions:

- Is the problem likely to be related to all the items in the list?
- If I eliminate one or two conditions, does the problem still occur?

Breaking Down the Essential Conditions

Once you have reduced the list, break down each essential condition into the essential conditions, using the same process. Once you seem to have the problem defined, test the definition one more time by asking the following questions:

- Have I considered all the pertinent data?
- Does the problem always occur when all the essential conditions are present?

When you feel confident the problem is defined, try to reproduce it. If it is reproducible, try to develop a test case for the benefit of the Unisys service engineer. If you cannot develop a test case, gather the media you think might help Unisys to reproduce the problem. These media include disks, tapes, and printouts pertaining to the problem.

Defining a problem can be a complex process. You must use your own judgment to determine how far to break down a problem. If you have spent a reasonable amount of time and still do not have the probable cause, stop and define the problem in terms of the essential conditions you have isolated. Then report the problem and let Unisys take over from there.

Reporting System Problems and Suggestions

Unisys customers report problems and suggest new features with a document called a User Communications Form (UCF). Typically, when you encounter a problem, you contact your local support center and the representative resolves your difficulty immediately. If your problem cannot be resolved immediately, the representative creates a UCF.

Creating a User Communication Form (UCF)

For the products included in an A Series software release, Unisys supplies a file called UCF/NOTES/<release level> on the DOCUMENTS media that accompanies the release. This file describes the information that is typically needed when you report problems.

When you report a problem to your customer representative, Unisys asks you to describe it as well as you can, and to supply additional information, such as program dumps and source listings, to help identify the problem and verify its correction. The information needed depends on the product or products involved and the type of problem.

Include or attach the following items with all UCFs you submit:

- The release levels of the compiler, operating system, and pertinent software.
- Identification of all patches that have been applied to the pertinent software beyond those included in the standard release, regardless of whether these patches are local or supplied by Unisys.

If you request, all machine-readable media except cassettes are returned to you. Each media item must have a label attached listing the return address and stating that the item is to be returned.

Handling the User Communication Form (UCF)

When your support center handles a UCF, the representative assigns a unique number to it and reports the number to you. You should use this number to identify the UCF both in your log book and in any inquiries regarding the UCF. When the UCF is added to the Unisys Engineering database, an acknowledgement copy of the UCF is returned to you.

You should maintain a log of the UCFs that are submitted to Unisys on your behalf. Keeping such a log serves several purposes:

- It aids you in identifying any outstanding UCFs.
- It aids in compiling the history of a particular problem.

A UCF that reports a problem is called a Trouble Report (TR). A UCF that suggests additional system capabilities or that requests that a product function differently is called a New Feature Suggestion (NFS).

The exact procedure that Unisys follows in handling your problem is dependent on the services for which you have contracted, and on where you are located. Please contact your local representative for more information about who can handle your problem.

Glossary

A

access structure
The central disk file that the system uses to determine the location of files stored on the disk subsystem. On noncataloging systems, the access structure contains entries that are used to locate files that are accessible to the system. On cataloging systems, the access structure contains entries that are used to locate all available versions of a file. The access structure is also called the catalog on both cataloging and noncataloging systems.

accesscode
An identification code, subordinate to a usercode, which sometimes has an associated password of its own. An accesscode can be specified in the USERDATAFILE as required along with a usercode/password combination. An accesscode further establishes a user's identity, controls security, and restricts access to disk files.

active
Pertaining to the state of a process that is executing normally, and is neither scheduled nor suspended.

active time
The period during which a processor, I/O channel, or both are devoted to a specific task. The active time is a composite of the processor and I/O time; it is not the sum of those two values.

Activity Reporting System (BARS) utility
A real-time utility program that monitors the performance of the system and displays performance statistics as numeric values and bar graphs. The items displayed are those concerning current Central Processor Unit (CPU), memory, I/O, and disk pack use.

actual segment
A group of contiguous words in main memory or on the disk subsystem. An actual segment can be a virtual segment or a single page of a paged virtual segment.

actual segment descriptor (ASD)
A pointer to the location of a data or code item in memory or on a disk.

ADM
See automatic display mode.

ALGOL
Algorithmic Language. A structured, high-level programming language that provides the basis for the stack architecture of the Unisys A Series systems. ALGOL was the first block-structured language developed in the 1960s and served as a basis for such languages as Pascal and Ada. It is still used extensively on A Series systems, primarily for systems programming.

alternate halt/load family
A family that contains a backup copy of the master control program (MCP) object code file and is not the current halt/load family. If the current halt/load family fails, the alternate halt/load family can be used to operate the system.

alternate halt/load unit
A disk that contains a duplicate master control program (MCP) object code file and is a member of a current halt/load family.

ancestor
The parent of a particular task, or the parent of any ancestor of the task.

ARCHIVING
The system option that, when enabled with the OP (Options) system command, enables the system to store and retrieve information about old generations of cataloged files in a database.

area
The amount of contiguous disk space that is allocated at one time to a disk file as it is being created or expanded.

ASD
See actual segment descriptor.

ASD memory
The memory architecture used on A Series systems. In this memory architecture, memory is treated as a single continuous region that is indexed by the ASD table. Memory management is very flexible and is handled automatically by the operating system.

ASD number
A number used as an index to the actual segment descriptor (ASD) table. The ASD number is contained in code descriptors and data descriptors.

ASD table
A memory-resident table that contains the actual segment descriptors (ASDs) for the system.

attribute
(1) A characteristic or property. (2) The information that describes a characteristic of an entity.

attribute name
The name of an attribute of a file, task, or database.

audit file
In data communications, a file that is produced for each network support processor (NSP) and data communications data link processor (DC-DLP) by the data communications subsystem procedures of the operating system. The audited items are specified with a Menu-Assisted Resource Control (MARC) menu selection or through an operator command.

automatic display mode (ADM)
A display mode that can be initiated at an operator display terminal (ODT) through the use of the ADM (automatic display mode) system command. In this mode, various types of information about the system are displayed at regular intervals.

available disk table
A table maintained by the system that keeps track of the space available on each disk in the system.

B

back up
To copy information to a disk or a tape to provide a means of restoring the information on the system as required.

backup file
(1) A printer or punch file assigned to a backup peripheral for subsequent output. The default backup peripheral is a disk. (2) A copy of a file that is stored offline so that it can be copied back onto the system if the original file becomes corrupted or inaccessible. (3) A copy of a file on a cataloging system that has been saved with one of the following Work Flow Language (WFL) statements: *COPY & BACKUP*, *ARCHIVE DIFFERENTIAL*, *ARCHIVE FULL*, *ARCHIVE INCREMENTAL*, or *ARCHIVE ROLLOUT*.

BARS
See Activity Reporting System (BARS) utility.

base pack
A disk that contains a copy of the system directory for the family of that disk and is currently being used by the system to identify and access the family.

batch job
A file that contains all the commands needed to accomplish a task or a series of tasks. A batch job is executed through Work Flow Language (WFL) or a Command and Edit (CANDE) command.

batch mode
(1) An execution mode in which a group of commands or other input is transmitted and processed by the computer with no user interaction. (2) In the Communications Management System (COMS), an execution mode in which a program running under COMS can do batch-type updates to a database shared by other transaction processors.

beginning of job (BOJ)
(1) The start of processing of a job. (2) In the Communications Management System (COMS) and X.25, the control code that signals the receiver that a job is beginning.

beginning of task (BOT)
The start of processing of a task.

BNA

The network architecture used on A Series, B 1000, and V Series systems as well as CP9500 and CP 2000 communications processors to connect multiple, independent, compatible computer systems into a network for distributed processing and resource sharing.

BOJ

See beginning of job.

BOT

See beginning of task.

C

cache memory

A mechanism interposed in the memory hierarchy between main memory and the central processor unit (CPU) to improve memory transfer rates and, hence, increase processor speeds.

calling process

A process that is linked to a library process and can import objects from that library process. *See also* user process.

CANDE

See Command and Edit.

card file

A file whose KIND attribute has the value READER.

catalog

(1) One of the sections in the catalog file. The catalog is a file that contains the following information for all files on the system: the available versions of a file, and the backup copies of the available versions of a file. (2) The central disk file that stores the information about the disks in the system and the disk files. This file is named SYSTEM/CATALOG/<family index number> on cataloging systems and SYSTEM/ACCESS/<family index number> on noncataloging systems.

catalog family

The disk family designated by the DL (Disk Location) system command as being the one on which the SYSTEM/CATALOG or SYSTEM/ACCESS file is stored.

catalog level

An integer value that determines how many generations of each file an installation can have. The catalog level is established by assigning a value to the CATALOGLEVELSET define that is compiled into the master control program (MCP). CATALOGLEVELSET can be assigned a value in the range 1 through 7. The default value is 3.

catalog rebuilding

The process on a cataloging system in which the system updates the file access structure table (FAST) with information about the location of catalog records for files on a disk family.

cataloged file
A file that has been entered into the catalog. The user can enter a file into the catalog by using the *CATALOG ADD* or *COPY & CATALOG* Work Flow Language (WFL) statement, by assigning to the USECATALOG file attribute the value TRUE, or by enabling the system option USECATDEFAULT.

CATALOGING
A system option that, when enabled with the OP (Options) system command, allows the system to keep track of copies of files that have been backed up onto a tape or a disk.

central processing unit (CPU)
The computer hardware unit that controls and executes the instructions contained in object code files.

chargecode
An account number to which the cost of a computer session is billed. System administrators can determine how chargecodes are used on their particular systems.

checkerboarding
A situation in which only small areas are available between the in-use areas of storage. Many areas can be available, but the system might not be able to use them because none of the areas is large enough.

checkpoint
A place in a program where the program is to be stopped so that its current state can be written to disk. After the state of the program has been recorded, program execution resumes where it left off. If the system halts unexpectedly before the program finishes, the program can be restarted at its most recent checkpoint instead of at the beginning.

class
In Work Flow Language (WFL), pertaining to an attribute for assigning jobs to a queue.

client process
A process that is linked to a library process and can import objects from that library process. *See also* user process.

code file
See object code file.

code segment dictionary
A memory structure that is associated with a process and that indexes the memory addresses of the various segments of program code used by that process. The same code segment dictionary can be shared by more than one process, provided that each process is an instance of the same procedure. A code segment dictionary is also referred to as a D1 stack.

cold-start
A procedure that can be used during system initialization that places the operating system on a disk and in memory. The system configuration is also defined to the operating system at this time. A cold-start reinitializes all data structures on a disk, causing any information about existing files on that disk to be lost.

Command and Edit (CANDE)
A time-sharing message control system (MCS) that enables a user to create and edit files, and develop, test, and execute programs, interactively.

Communications Management System (COMS)
A general message control system (MCS) that controls online environments on A Series systems. COMS can support the processing of multiprogram transactions, single-station remote files, and multistation remote files.

complementing
The process the system uses to construct an available disk table. The operating system starts with the entire available disk and then reads the disk file headers on a family to determine the space that is already allocated. The operating system then takes the complement of the allocated space to determine the space that is still available.

COMS
See Communications Management System.

configuration
(1) A description of selected hardware or software options or capabilities. (2) A set of hardware resources in an installed system. In particular, a configuration is a representation of the devices in the I/O subsystem and memory subsystem.

configuration file
(1) A table that contains the configuration of a system. The configuration table is stored in the disk directory of the halt/load family. (2) For the SYSTEM/CONFIGURATOR utility, a file that lists and describes the hardware resources and selected software information that make up a configuration. The configuration file can contain descriptions of several different hardware and software configurations for a system. (3) In the Communications Management System (COMS), a file that contains descriptions of the tables defined through the COMS Utility program. These tables contain information on message routing, security, dynamic program control, and synchronized recovery. This file is also referred to as the COMS CFILE.

contention
(1) The situation in which multiple users, such as tasks, compete for shared resources, such as processors, I/O devices, and records in a file. (2) In data communications, the situation in which multiple users vie for the right to use a transmission channel within a multiplexed digital facility. The condition arises when either of the following is true: two or more data terminal installations attempt to transmit at the same time over a shared channel; or two data terminal installations attempt to transmit at the same time in two-way alternate or two-way simultaneous data communication.

continuation pack
A disk that is not currently being used as the base pack for a multipack family. A continuation pack can have a copy of the system directory for the family.

control command
A Command and Edit (CANDE) input message that begins with a specific character, usually a question mark (?), and is used to control or interrogate the CANDE operating environment. A control command can be entered from any attached CANDE station.

control program (CP)
A program that has been marked by the CONTROL option of the MP (Mark Program) system command. A control program runs at a higher priority than ordinary tasks and is less likely to be held scheduled.

control station
(1) In Command and Edit (CANDE), a station that allows CANDE network control commands to be entered. (2) In the Communications Management System (COMS), a control-capable station; that is, a station with no restrictions on the use of COMS commands. A control station is defined in one of three ways: through the COMS Utility, in the Network Definition Language II (NDLII), or through the Interactive Datacomm Configurator (IDC) to set its station attribute SPO to TRUE.

CONTROLLER
An invisible, independent runner program that is responsible for processing system commands, routing system messages, and scheduling jobs.

CP
See control program.

CPU
See central processing unit.

crunch
To return the unused portion of the last row of disk space of a disk file (beyond the end-of-file indicator) to the system. Only disk files can be crunched.

CYCLE
A file attribute that can be used with the VERSION file attribute to distinguish the generations of a file.

D

DASDL
See Data and Structure Definition Language.

Data and Structure Definition Language (DASDL)
In Data Management System II (DMSII), the language used to describe a database logically and physically, and to specify criteria to ensure the integrity of data stored in the database. DASDL is the source language that is input to the DASDL compiler, which creates or updates the database description file from the input.

data comm
See data communications.

data communications (data comm)
The transfer of data between a data source and a data sink (two computers, or a computer and a terminal) by way of one or more data links, according to appropriate protocols.

Data Communications ALGOL (DCALGOL)
A Unisys language based on ALGOL that contains extensions for writing message control system (MCS) programs and other specialized system programs.

data communications data link processor (DCDLP)
A data communications processor (DCP) that combines the functions of a network support processor (NSP) and a line support processor (LSP) into one physical data link processor (DLP) and supports up to four lines of communication.

data link processor (DLP)
A processor that serves as the system interface to a specific peripheral device, controller, or communications network.

Data Management ALGOL (DMALGOL)
A Unisys language based on ALGOL that contains extensions for writing Data Management System II (DMSII) software and other specialized system programs.

Data Management System II (DMSII)
A specialized system software package used to describe a database and maintain the relationships among the data elements in the database.

database stack (DBS)
A stack that contains all the information necessary for the Data Management System II (DMSII) Accessroutines to manage a database.

DATACOMINFO file
A file that contains a complete description of the data communications configuration, including algorithms, editors, and translate tables. This is the file that the Interactive Datacomm Configurator (IDC) modifies and from which the operating system initializes the data communications subsystem.

DBS
See database stack.

DCALGOL
See Data Communications ALGOL.

DCDLP
See data communications data link processor.

default
Pertaining to a value automatically assigned by a program or system when another value has not been specified by the user.

default printer pool
A group of printers used for print requests with unspecified destinations.

default value
The value automatically given to a variable when no other value has been assigned.

dependent process
A process that depends on the continued existence of another process, called the parent process. *See also* task.

descendant
An offspring of a particular process, or an offspring of a descendant of that process.

device
Any piece of I/O hardware, such as a data link processor (DLP) or a peripheral.

direct printer file
A file that is routed directly to the printer.

direct window
In the Communications Management System (COMS), a type of window that enables the user to route messages directly to COMS, while using all the COMS capabilities for preprocessing and postprocessing of messages.

directory
(1) A table of contents listing the files contained on a device. The device is usually a disk or a tape. (2) A list of file names organized into a hierarchy according to similarities in their names. File names are grouped in a directory if their first name constants (and associated usercodes) are identical. These groups are divided into subdirectories consisting of those file names whose first two name constants are identical, and so on. (3) Special disk files used by the system that include archive directories, catalogs, and system directories.

discontinue
(1) To terminate a referenced task. (2) To cause a process to terminate abnormally. A process can be discontinued by system commands, by statements in related processes, or by the system software.

disk
A random-access data storage device consisting of one or more circular platters that contain information recorded in concentric circular paths called tracks. Data on a disk are accessed by movable read/write heads. Some disks are removable. *Synonym for* disk pack, pack.

disk drive
The device on which a disk is mounted. The disk drive has movable read/write heads that access the data on the disk.

disk drive controller
The device that controls the disk drive units and transfers information between the host system and the disk drive units. On some systems, this device is also referred to as a D-machine.

disk file header
A data structure that contains information about a disk file, such as the physical location of the file on the disk and various file attributes. A disk file header is also referred to as a header.

disk pack
A random-access data storage device consisting of one or more circular platters that contain information recorded in concentric circular paths called tracks. Data on a disk pack are accessed by movable read/write heads. Some disk packs are removable. *Synonym for* disk, pack.

DMALGOL
See Data Management ALGOL.

DMSII
See Data Management System II.

dump
(1) A copy of data in memory or a copy of a program on a tape. (2) A process that writes the contents of a specific area of memory to record the contents of a particular stage of processing. This process is used to determine the cause of a failure. (3) To copy a file or program on a tape. (4) In data or file management, a copy of a database on a tape.

E

elapsed limit
A limit on the elapsed time that a task is allowed.

elapsed time
The actual time that has passed since a particular event occurred, such as since a task was initiated. Elapsed time is also referred to as wall-clock time.

enabled
Referring to a station that is being polled (invited to transmit in a certain order) and that can communicate with the system.

end of job (EOJ)
The termination of processing of a job.

end of task (EOT)
The termination of processing of a task.

end-of-text character (ETX)
A keyboard character used to signal the end of input.

entry point
A procedure or a function that is a library object.

EOJ
See end of job.

EOT
See end of task.

ETX
See end-of-text character.

execution
The act of processing statements in a program.

execution time
The time during which an object code file is executed. *Synonym for* run time.

F

family
(1) One or more disks logically grouped and treated as a single entity by the system. Each family has a name, and all disks in the family must have been entered into the family with the RC (Reconfigure Disk) system command. (2) The name of the disk or disk pack on which a physical file is located.

family index
A 3-digit number the system assigns to a disk when the disk is added to a family. This family index value must be in the range 1 to 255, inclusive. The base pack is assigned the family index number, the first continuation pack is assigned 002, and so on. A family index is also referred to as a family index number.

family index number
See family index.

family member
A disk that is a base or continuation pack in a family.

family name
(1) The name, consisting of up to 17 alphanumeric characters, assigned by an installation to identify a family of disks. (2) The name (label) of the disk or disk pack on which a physical file is located. The family name of a file is determined by the value of the FAMILYNAME file attribute. (3) The name of the logical group of disk packs on which a physical file is located. A family name consists of from 1 to 17 alphanumeric characters and is assigned by the installation.

family rebuild
The process in which the system reconstructs the file access structure table (FAST) entry for a family by reading its flat directory.

family substitution
A method for redirecting references to files on a disk family to avoid entering the actual family name in commands or file names. For example, if a user enters *FAMILY DISK=PACK OTHERWISE DISK*, that user's file requests are checked on disk packs named PACK and DISK.

FAST
See file access structure table.

FETCH specification
A statement in a Work Flow Language (WFL) job that provides a message an operator can display with a PF (Print Fetch) system command. Resetting the NOFETCH system

option delays initiation of jobs with FETCH specifications until the operator enters an OK command for each job.

file
A named group of related records.

file access structure table (FAST)
Special records in the access structure or catalog directory that the system uses to locate disk files. The FAST contains a pointer to the header of each disk file in the system directory of each family.

file attribute
An element that describes a characteristic of a file and provides information the system needs to handle the file. Examples of file attributes are the file title, record size, number of areas, and date of creation. For disk files, permanent file attribute values are stored in the disk file header.

file name
(1) A name or word that designates a set of data items. (2) A unique identifier for a file, consisting of name constants separated by slashes. Each name constant consists of letters, digits, and selected special characters. A file name can be optionally preceded by an asterisk (*) or usercode, and optionally followed by ON and a family name.

file title
The complete identifier for a file that consists of the file name, the word ON, and the family name.

file transfer
The communication of files between host systems, workstations, or terminals. The category of distributed systems service (DSS) that enables a user to transfer files between host systems, workstations, or terminals. *See also* library maintenance.

flat directory
See system directory.

G

genealogy
The ranking of a generation of a file relative to the other generations of that file. The generation with the highest CYCLE file attribute, the highest VERSION file attribute within that CYCLE, and the most recent timestamp within that CYCLE and VERSION is said to have the best genealogy.

generation
A particular, available copy of a file. The generation of a file is determined by the file attributes CYCLE and VERSION, and the timestamp of the file.

GENERATION
A file attribute on cataloging systems that enables the user to select a particular generation of a file.

GETSTATUS
An intrinsic routine available to some programming languages that can be used to obtain the status of the job or task mix, the status of peripheral and disk units, and the status of the operating system and mainframe configuration.

guard file
A disk file created by the GUARDFILE utility program that describes the access rights of various users and programs to a program, data file, or database.

H

halt/load
A system-initialization procedure that temporarily halts the system and loads the operating system from a disk to main memory.

halt/load family
The disk family that contains the currently operative master control program (MCP) object code file.

halt/load unit
(1) The disk pack that contains the currently operative master control program (MCP) object code file. (2) The disk drive unit on which the halt/load disk is mounted.

header
(1) A data structure that contains information about a disk file, such as the physical location of the file on the disk and various file attributes. A header is also referred to as a disk file header.

Host Services
A collection of services that are provided across multihost networks of A Series, B 1000 Series, CP9500, and V Series systems. The services can include file transfer, job transfer, station transfer, logical I/O, remote tasking, and remote command entry.

I

I/O
Input/output. An operation in which the system reads data from or writes data to a file on a peripheral device such as a disk drive.

I/O control block (IOCB)
A data structure used for communication between the host system and the I/O subsystem.

I/O control word (IOCW)
The area in the I/O control block (IOCB) where information about the I/O operation to be performed resides.

I/O controller
The processor that provides the interface between the host and a peripheral device such as a disk drive controller.

I/O processor (IOP)
A specialized processor for moving data between system memory and the I/O subsystem.

image printer
A printer that prints a page of images at one time, as opposed to a line printer, which prints a line of images at one time.

in-use process
A process that has been submitted for initiation and has not yet terminated. The state of an in-use process can be scheduled, active, or suspended.

independent process
A process that does not depend on the continued existence of a parent process. An independent process is the head of any process family it is part of. *See also* job.

independent runner
A master control program (MCP) procedure that is initiated as an independent process. The procedure is executed in its own process stack rather than in the stack of a user process. An independent runner can be either visible or invisible. If the independent runner is visible, its status can be interrogated. If the independent runner is invisible, it does not appear in mix displays.

indirect output
The output from a program that is not sent directly to a printer. In most cases, the output is sent to a backup file.

InfoGuard
The Unisys security-enhancement software for A Series systems. InfoGuard provides such features as password management, selective logging and auditing, tape volume security, and simplified system-security configuration.

inheritance
The automatic transfer of particular task attribute values from a process to a descendant process. More broadly, inheritance also refers to the automatic transfer of values from job queue attributes or session attributes to the equivalent task attributes of a descendant process.

initialization
(1) The process of starting a program and giving starting values to variables. (2) A procedure that makes a system or subsystem available for its intended use. Important phases of initialization are the recognition of the physical environment, the identification of the available resources, and the establishment of the interface with the user. System initialization occurs as part of a halt/load.

initialization, verification, relocation (IVR)
A maintenance procedure used to write sector boundaries and a blank label on a disk. You can use the IVR procedure to make a new disk pack usable by the system or to make a damaged disk reusable by eliminating defective sectors. The end product of an IVR procedure is a master available table (MAT) of available disk segments with all old files on the disk erased.

initiation

A type of procedure invocation that causes the creation of a new process, with its own process stack and process information block (PIB). Additionally, a new code segment dictionary is created if a code segment dictionary for that procedure is not already available.

input device

A device from which data are read into the system. A card reader is always an input device. Other devices such as disks, disk packs, magnetic tape units, and terminals are input devices when data go from the device to the host.

interactive process

A process that reads input from a terminal or operator display terminal (ODT), and whose actions are largely determined by the input received. A data entry process, such as the Editor, is an example of an interactive process.

intrinsic

A system-supplied program routine for common mathematical and other operations that is loaded onto the system separately. An intrinsic can be invoked by the operating system or user programs.

IOCB

See I/O control block.

IOCW

See I/O control word.

IOP

See I/O processor.

IVR

See initialization, verification, relocation.

J

job

An independent process. The job of a particular task is the independent process that is the eldest ancestor of that task.

job description file

A system disk file that stores information about Work Flow Language (WFL) jobs, job queues, and various system settings. The job description file is also known as the JOBDESC file.

job file

A disk file that is associated with a job and contains the job log. The job file for a Work Flow Language (WFL) job also serves as the object code file for the job, and includes job restart information, data specifications, and a copy of the WFL source program.

job log
A log that is stored in a job file and contains log entries for a particular job and its descendant tasks. When the job terminates, the job log is processed to produce the job summary.

job queue
A structure in the system software that stores a list of jobs that have been compiled and are waiting to be initiated.

job queue number
A unique identifier for a particular job queue.

job summary
A file, produced after a job completes execution, that lists information such as the tasks initiated by the job, the beginning and ending times for each task, and the termination information for each task.

L

label
(1) The first 28 sectors on a disk, on which information about the disk is stored. This information includes the family name and serial number, the master available table (MAT), the family index number, information about the family base pack, and a pointer to the system directory if the disk contains a directory. (2) An area on a magnetic tape (MT) that contains permanent attributes associated with the tape volume or with individual files on the volume, such as the volume serial number and the file name.

LAST
See local access structure table.

library
(1) A collection of one or more named routines or entry points that are stored in a file and can be accessed by other programs. (2) A program that exports objects for use by user programs.

library maintenance
A master control program (MCP) procedure that copies disk files to and from disk, tape, and compact disk (CD). Library maintenance is invoked by the Work Flow Language (WFL) statements *ADD, COPY,* and *ARCHIVE. See also* file transfer.

library maintenance tape
A tape created by library maintenance that contains backup copies of disk files.

library object
An object that is shared by a library and one or more user programs.

library process
An instance of the execution of a library. The sharing option of a library determines whether multiple user programs use the same instance of the library.

line printer (LP)
A peripheral device that prints all characters of a line as a unit.

line support processor (LSP)
The data communications subsystem processor that manages communication with the host and initiates processes that control the input of messages to and the output of messages from data communications lines.

LOADER
A stand-alone program loaded into main memory and then executed. LOADER is used primarily to create a halt/load pack on a system when the operating system cannot run. This condition usually occurs when the last available halt/load pack has been destroyed.

local access structure table (LAST)
A special file located on the base pack that is used to update the file access structure table (FAST) each time a base pack is brought online. The main value of the LAST is that a time-consuming family rebuild is not necessary when a disk is readied on a noncataloging system.

log file
A file that contains a record of system activity, including system utilization, messages, and peripheral activity.

log station
A Command and Edit (CANDE) station designated to receive logging information. This information can include statistics on station attachment, security errors, station log-on and log-off times, network changes caused by reconfiguration requests, and user messages sent to the operator display terminal (ODT).

logging
The process of recording events and, often, their times of occurrence.

LP
See line printer.

LSP
See line support processor.

M

MAKEUSER
A utility used to define, modify, or display information about the usercodes that are available on the system. The usercode information is stored in a file called the USERDATAFILE.

MARC
See Menu-Assisted Resource Control.

Mark release
A Unisys A Series system software release identified by a release level number. Releases are periodic updates to system software. When a major change in the capability or

design of a product is made, the release level number increases by 0.1.0. For changes between major releases, the number increases by 0.0.1.

master available table (MAT)
A table stored on each disk that lists the valid sectors on the disk that were successfully processed by the initialization, verification, relocation (IVR) procedure. Pointers to defective sectors are deleted from the MAT so that these sectors cannot be accessed. Normally, the MAT shows the entire disk as being available, minus any defective sectors.

master control program (MCP)
The central program of the A Series operating system. The term applies to any master control program that Unisys may release for A Series systems.

MAT
See master available table.

MCP
See master control program.

MCS
See message control system.

Memory Disk
A Unisys software feature that enables the use of memory as if it were a disk unit and provides file access with extremely high data-transfer rates and relatively little access time.

memory dump
A copy of the contents of memory, sometimes referred to as a memory image. The system produces a memory dump when a problem occurs so that the problem can be analyzed.

Menu-Assisted Resource Control (MARC)
A menu-driven interface to A Series systems that also enables direct entry of commands.

message control system (MCS)
A program that controls the flow of messages between terminals, application programs, and the operating system. MCS functions can include message routing, access control, audit and recovery, system management, and message formatting.

mirror information table (MIT)
A system status table that contains information about all mirrored sets on the system. The MIT is saved in the directory of the halt/load disk. After a halt/load, the MIT is read into memory from the directory of the halt/load disk.

Mirrored Disk
A software feature on Unisys A Series systems that enables from two to four disks to be maintained as a mirrored set; that is, as exact copies of each other. The Mirrored Disk feature increases system availability and data integrity, decreases the possibility of data loss due to equipment or media malfunction, and can improve I/O throughput on disk subsystems that experience a high ratio of reads versus writes.

MIT

See mirror information table.

mix

The set of processes that currently exist on a particular computer. The mix can include active, scheduled, and suspended processes.

mix number

A 4-digit number that identifies a process while it is executing. This number is stored in the MIXNUMBER task attribute.

multidisk family

A family that consists of more than one disk. The system treats the family as a single entity.

N

NDLII

See Network Definition Language II.

Network Definition Language II (NDLII)

The Unisys language used to describe the physical, logical, and functional characteristics of the data communications subsystem to network support processors (NSPs), line support processors (LSPs), and data communications data link processors (DCDLPs).

network support processor (NSP)

A data communications subsystem processor that controls the interface between a host system and the data communications peripherals. The NSP executes the code generated by the Network Definition Language II (NDLII) compiler for line control and editor procedures. An NSP can also control line support processors (LSPs).

NEWP

A structured, high-level programming language used in developing some Unisys system software. Based on the ALGOL language, NEWP contains facilities necessary for the operating system to interact with A Series hardware.

nonremovable disk

A disk that cannot be removed from the disk drive on which it is mounted.

nonresident file

A file stored on a backup tape, or a backup copy of a file that is stored on a different disk family from that on which the primary copy of the file is stored. This term does not always pertain to the RESIDENT file attribute.

NSP

See network support processor.

null mix

A situation in which no processes currently exist on the system, except system software processes such as master control program (MCP) independent runners.

O

object code
The instructions in machine code that are created as a result of compiling source code.

object code file
A file produced by a compiler when a program is compiled successfully. The file contains instructions in machine-executable object code.

object program
A set or group of executable machine-language instructions and other material designed to interact with data to provide problem solutions. An object program is generally the machine-language result of the operation of a high-level language compiler on a source program. *See also* object code file.

ODT
See operator display terminal.

ODT command
An obsolete term; *see* system command.

offline
(1) Pertaining to the state of being incapable of immediate communication with a central computer. (2) Pertaining to the state of not being accessible by the operating system. (3) In the database environment, pertaining to the state of not being available to users.

online
(1) Pertaining to the state of being capable of immediate communication with a central computer. (2) Pertaining to the state of being accessible by the operating system. (3) Pertaining to a disk mounted on a drive that is ready and has not been made inaccessible to the system with the UR (Unit Reserved), SV (Save), CLOSE (Close Pack), or FREE (Free Resources) system command, and whose label and system directory (if it has one) have been read successfully. (4) In the database environment, pertaining to the state of being available to users.

operation
An action undertaken or executed by a computer. Addition, multiplication, extraction, comparison, and transfer are examples of operations.

operator display terminal (ODT)
(1) A terminal or other device that is connected to the system in such a way that it can communicate directly with the operating system. The ODT allows operations personnel to accomplish system operations functions through either of two operating modes: system command mode or data comm mode. (2) The name given to the system control terminal (SCT) when it is used as an ODT.

output device
See device.

outstanding write list (OWL)
A table used by a disk subsystem that uses the Mirrored Disk feature. The table is used to ensure that data written to one disk in a mirrored set is also written to the remaining disks in the set.

overlay
To load code or data into a memory area that was previously allocated to other code or data, and to write any data that previously occupied the area to a disk file if necessary.

OWL
See outstanding write list.

P

pack (PK)
A random-access data storage device consisting of one or more circular platters that contain information recorded in concentric circular paths called tracks. Data on a pack are accessed by movable read/write heads. Some packs are removable. *Synonym for* disk pack, disk.

pack access structure table (PAST)
A table used by the system to locate disk families. The PAST contains pointers into the file access structure table (FAST) that indicate where the entries for the files in a family are stored.

password
A character string associated with a usercode or accesscode in the USERDATAFILE, and used to identify legitimate users of the system. When logging on to a message control system (MCS), a user must supply a usercode and a password.

PAST
See pack access structure table.

performance
(1) A measurement of how efficiently a process uses resources such as processor time, I/O time, or elapsed time. (2) A measure of the amount of work a computer system is able to do in a given period of time.

peripheral
A device used for input, output, or file storage. Examples are magnetic tape drives, disk drives, printers, or operator display terminals (ODTs). *Synonym for* peripheral device.

peripheral device
A device used for input, output, or file storage. Examples are magnetic tape drives, printers, disk drives, and operator display terminals (ODTs). *Synonym for* peripheral.

physical unit number
The permanent identification for a device by which it is known to the system.

preinitialized disk
A disk the factory has prepared for use on the system. The preparation includes initialization, verification, and relocation of any defective sectors, as well as labeling of the disk. The required files are copied onto the disk, including the BOOTCODE and operating system files. Finally, the disk is designated as the halt/load unit and selected as the Boot unit.

print request
(1) A request that contains information describing the location, time, and method of printing a printer file or group of printer files. (2) A group of one or more files to be printed, along with the attributes that describe when, where, and how to print them.

print request list
A list containing information about all outstanding print requests. Outstanding print requests include those that are queued for printing, those with exceptions, those that are waiting for resources, and those scheduled for printing selection in the future.

print servers
In the print system, a stack or process that performs the output to a printer device.

Print System (PrintS)
A Unisys software product used to control when, where, and how printer backup files are printed on A Series systems.

printer backup file
A system-formatted file that contains data to be printed and carriage control information. A printer backup file refers to both printer and card punch output.

printer pool
See default printer pool.

PrintS
See Print System.

priority
A characteristic associated with a process that determines its precedence in the use of system resources. A process with higher priority executes more quickly than it would if it had lower priority.

private file
A file with a SECURITYTYPE attribute specified as PRIVATE. Only users logged on to a privileged usercode or to the usercode under which the file is stored can access a private file.

private process
A process whose task attributes cannot be accessed by other processes. Assigning the *private process* option to the OPTION task attribute causes a process to become a private process.

privilege
The ability to invoke actions that are not ordinarily allowed, such as accessing private files stored under other usercodes or invoking privileged functions such as SETSTATUS. The concept of privilege applies to usercodes, programs, and processes.

privileged process
A process that bypasses normal system security checks, thereby gaining access to a large number of system resources. A process initiated by a privileged program or run under a privileged usercode is a privileged process.

privileged program
An object code file marked as privileged with the PP (Privileged Program) system command.

privileged user
A user with the PU usercode attribute assigned to his or her usercode in the USERDATAFILE. No file-access security checking is normally performed for actions taken under a usercode with privileged status.

process
(1) The execution of a program or of a procedure that was initiated. The process has its own process stack and process information block (PIB). It also has a code segment dictionary, which can be shared with other processes that are executions of the same procedure. (2) A software application; that is, any activity or systematic sequence of operations that produces a specified result.

processor time
The accumulated time a task has utilized a central processor unit (CPU) or central processing module (CPM) resource. This excludes time spent waiting for I/O operations or waiting for other events.

public file
A file with a SECURITYTYPE attribute specified as PUBLIC. Users who are logged on to any usercode can access a public file by specifying the (<usercode>)<file name> form of the file title.

Q

queue
A data structure used for storing objects; the objects are removed in the same order they are stored. *See* job queue, ready queue.

queue attribute
Any attribute that can be assigned to a job queue using the MQ (Make or Modify Queue) system command. Queue attributes limit the use of system resources by jobs and tasks initiated from that queue. MIXLIMIT, PROCESSTIME, and PRIORITY are are examples of queue attributes.

R

railroad diagram
A graphic representation of the syntax of a command or statement.

ready queue
A list, maintained by the operating system, of the processes that are waiting for service from a processor.

rebuild
(1) In the disk subsystem, a concept that refers to either of the following: a family rebuild, in which the system constructs the file access structure table (FAST) entry for a family by reading its system directory; or a catalog rebuild (on a cataloging system), in which the system updates the file access structure table (FAST) entry with information about cataloged files. (2) In database management, a recovery process in which the entire database is loaded from one or more sets of dump tapes. The recovery process then applies the audit trail after-update record images to move the database forward in time.

remote device
An I/O unit or other piece of equipment that is physically removed from the computer but connected by a communications line.

remote file
A file with the KIND attribute specified as REMOTE. A remote file enables object programs to communicate interactively with a terminal.

Remote Print System (ReprintS)
A Unisys software system that controls the routing and printing of backup files at remote (data comm) destinations and on BNA networks.

removable disk
A disk that can be removed from the drive unit on which it is mounted.

ReprintS
See Remote Print System.

resident file
The primary copy of the file that is stored on a disk, regardless of whether or not the disk is online. Backup copies of files stored on another disk family are not considered resident. This term does not always pertain to the RESIDENT file attribute.

row
The amount of contiguous disk space that is allocated at one time to a disk file as it is being created or expanded.

resident program
See terminate and stay resident (TSR) program.

RSVP message
A message that the system displays for a suspended process that states the reason the process was suspended. RSVP messages ask for a reply such as *OK* or *DS*.

run time
The time during which an object code file or user interface system (UIS) is executed. *Synonym for* execution time.

S

save memory
An area of memory that cannot be overlaid as long as the item with which it is associated is allocated.

schedule queue
A list of all processes waiting to enter the mix.

scheduled process
A process whose initiation is delayed, either because an operator has entered an HS (Hold Schedule) system command or because the operating system estimates the process is likely to need more memory than is currently available.

scratch tape
A labeled magnetic tape (MT) whose label indicates that there are no files on the tape. Old data might remain on the tape, but this old data cannot be read unless the tape is read as an unlabeled tape. The old data present on a scratch tape is written over when new data is written to the tape.

sector
A subdivision of a track on a disk. A sector is the minimum addressable area on a disk pack. Unisys A Series system sectors are 30 words, or 180 bytes, long.

security-administrator status
The status required to perform security administrator functions on systems with the system SECADMIN option set to TRUE; that is, the system primitive command *??SECAD* + has been designated. Security-administrator status is granted to a usercode by assigning it the SECADMIN usercode attribute in the USERDATAFILE.

segment
(1) *Synonym for* sector. (2) A contiguous region of memory that is referred to by a descriptor and stores code or data for use by a process.

segment dictionary
See code segment dictionary.

Semantic Information Manager (SIM)
The basis of the InfoExec™ environment. SIM is a database management system used to describe and maintain associations among data by means of subclass-superclass relationships and linking attributes.

InfoExec is a trademark of Unisys Corporation.

serial number
The 6-character field an installation assigns to a disk or magnetic tape to uniquely identify it. The serial number is stored on the label of the disk or tape.

session
(1) The interactions between a user and a message control system (MCS) during a particular period of time that is assigned an identifying session number. Logging on initiates a new session; logging off terminates a session. Each Menu-Assisted Resource Control (MARC) or Command and Edit (CANDE) dialogue at a terminal accesses a different session. (2) In the Command and Edit (CANDE) message control system (MCS), the time measured from when a CANDE user enters a valid usercode/password combination until that user enters a CANDE *SPLIT*, *BYE*, or *HELLO* command. Each CANDE session is assigned a unique number. No output is printed until the session is ended.

SI
See Simple Installation.

SIM
See storage interface module, Semantic Information Manager.

Simple Installation (SI)
A program that simplifies the installation of Unisys system software through either batch commands or menu screens.

SMFII
See System Management Facility II.

spooling
The process of indirectly sending output files to peripheral devices such as printers. The output file is logically written to the peripheral device, but actually is sent to a tape or disk file known as a backup file. The backup file is written to the peripheral device when the job finishes. Spooling enables many users to share peripherals efficiently by preventing any one job from monopolizing a peripheral.

stack
A region of memory used to store data items in a particular order, usually on a last-in, first-out basis.

storage interface module (SIM)
A subdivision of memory storage on A 12 and A 15 systems. A SIM controls from one to four memory storage units (MSUs) on A 12 systems and from one to eight MSUs on A 15 systems.

SUMLOG
A comprehensive event-log file, created and maintained by the system, that contains information on resource usage, hardware errors, security violations, system messages, and job and file activity.

supervisor program
A customer-written program that has been designated as the first program to be entered in the mix after a halt/load.

support library
A library that is associated with a function name. User programs can access a support library by way of its function name instead of its object code file title. The operator uses the SL (Support Library) system command to link function names with libraries.

suspended process
A process that has temporarily stopped executing and cannot continue until appropriate operator or programmatic action is taken. A process can be suspended deliberately by an operator command or a statement in a program. In addition, the operating system can suspend a process automatically, for example, if the process has requested a file that is missing.

system command
Any of a set of commands used to communicate with the operating system. System commands can be entered at an operator display terminal (ODT), in a Menu-Assisted Resource Control (MARC) session, or by way of the DCKEYIN function in a privileged Data Communications ALGOL (DCALGOL) program.

system control terminal (SCT)
(1) A terminal or other device that is connected to the system in such a way that it can communicate directly with the maintenance processor. An SCT can operate in maintenance mode or in operator display terminal (ODT) mode. On some systems, the SCT also provides a remote support mode. (2) A terminal used to enter information. An SCT can be used three ways: as an operator display terminal (ODT) to interface with the operating system, as a maintenance display terminal (MDT) to interface with the maintenance subsystem, or as a remote display terminal (RDT) to interface with remote support. The windows providing these uses are available once the automatic initialization sequence has finished.

system directory
(1) A special disk file on each disk family that the system uses to locate files on that family. The system directory, also referred to as the flat directory, contains a disk file header for each permanent file in the family. (2) The logical directory for nonusercoded files.

system file
A file with a special security status that protects it from being removed, retitled, or replaced except by selected system interfaces. For example, the job description (JOBDESC) file can be removed only by the ??RJ (Remove JOBDESC File) system command.

system library
A library that is part of the system software and is accorded special privileges by the operating system. Two examples of system libraries are GENERALSUPPORT and PRINTSUPPORT.

system log file
A file that contains a record of events for a particular system.

System Management Facility II (SMFII)
A software system that monitors and provides data on four aspects of system performance: hardware, software, workload characterization, and system utilization.

system software
The master control program (MCP) and all other object code files necessary for system operation.

SYSTEM/ACCESS
The name of the file on noncataloging systems that contains the access structure.

SYSTEM/CATALOG
The name of the file on cataloging systems that contains the access structure and the catalog.

SYSTEMUSER
(1) A type of user status that allows the user access to BNA network commands and to system commands through Menu-Assisted Resource Control (MARC) and through the BNA operator display terminal (ODT) Distributed Systems Services (DSS). However, the following are two exceptions to this access: system primitive commands and those commands reserved for users with security-administrator status when that status is authorized. SYSTEMUSER status is granted to a usercode by making the SYSTEMUSER attribute TRUE for the usercode in the USERDATAFILE. (2) A user with SYSTEMUSER status.

T

tape drive
An I/O peripheral device that stores data on reels or cartridges of magnetic tape (MT).

task
(1) A dependent process. (2) Any process, whether dependent or independent. *See also* process.

task attribute
Any of a number of items that describe and control various aspects of the execution of a process. Various operator commands, message control system (MCS) commands, and programming language statements can be used to interrogate and modify the values of task attributes. The task attributes of a process are stored in the process information block (PIB).

task file
A printer backup file that is associated with each process, and that stores any program dumps generated by the process while the TOPRINTER program dump option is enabled. Processes can also write comments to the task file by way of the TASKFILE task attribute.

tasking
The act of initiating, monitoring, or controlling processes. The processes can be either jobs or tasks. Operators and users can enter tasking commands from an operator display terminal (ODT), a Command and Edit (CANDE) session, or a Menu-Assisted Resource Control (MARC) session. Programs can initiate processes with such statements as CALL, PROCESS, or RUN. Programs can monitor and control processes by reading and assigning the values of various task attributes.

terminate and stay ready (TSR) program
A program that has been loaded into the random-access memory (RAM) of a personal computer (PC) and that typically stays there until the PC is turned off or rebooted. When a TSR program occupies a section of memory, it is possible to run other programs in the unused part of memory. The TSR program remains available for immediate use without having to be loaded again from disk.

thrashing
A state of degraded system performance caused by overcrowding of main memory. The overcrowding of memory causes the system to spend an excessive amount of time performing overlays.

throughput
The total useful information processed during a specified time period.

timestamp
An encoded, 48-bit numerical value for the time and date. Various timestamps are maintained by the system for each disk file. Timestamps note the time and date a file was created, last altered, and last accessed.

trancode
See transaction code.

transaction code (trancode)
(1) A sequence of characters included in a message that indicates the agenda to apply to a message during preprocessing or postprocessing. (2) In the Communications Management System (COMS), a code that can appear in a transaction-initiating message header, indicating the processing that is to be carried out. This code is used to route the message to the appropriate host program.

transform function
In the Print System (PrintS), a general-purpose function that can manipulate a print line before it is printed on a device. An example is translating lowercase characters to uppercase before printing.

turnaround
The time required for a job to be completed. In dealing with job queue maintenance, turnaround time is the time that has elapsed since the last job was introduced into the mix from a particular queue.

U

UCF
See User Communication Form.

unit number
A number assigned by an installation to a peripheral device, such as a disk drive, and used to identify the device. The unit number commonly appears in conjunction with an acronym indicating the type of unit, which provides a unique identifier for a particular peripheral.

USECATDEFAULT
The system option that, when enabled with the OP (Options) system command, assigns TRUE as the default value of the USECATALOG file attribute.

User Communication Form (UCF)
A form used by Unisys customers to report problems and express comments about Unisys products to support organizations.

user process
(1) A process that is not an invisible independent runner, a message control system (MCS), or a system library. (2) A process that is linked to a library process and can import objects from that library process. *Synonym for* calling process, client process.

user program
(1) A program that is not part of the system software. (2) A program that uses objects imported from a library program.

usercode
An identification code used to establish user identity and control security, and to provide for segregation of files. Usercodes can be applied to every task, job, session, and file on the system. A valid usercode is identified by an entry in the USERDATAFILE.

usercode attribute
A characteristic that can be associated with a usercode in the USERDATAFILE. A set of standard usercode attributes, such as PU, MAXPW, IDENTITY, and PASSWORD, are supplied as part of the description of the USERDATAFILE structure. The system administrator or security administrator can define additional usercode attributes to meet the specific needs of an installation.

USERDATAFILE
A system database that defines valid usercodes and contains various data about each user (such as accesscodes, passwords, and chargecodes) and the population of users for a particular installation.

utility
A system software program that performs commonly used functions.

UTILOADER
A stand-alone program loaded into main memory and executed by the E-Mode Processor (EMP). UTILOADER is primarily responsible for the loading and execution of other stand-alone programs such as LOADER and BOOTSTRAP, or for creating the resident bootstrap of the master control program (MCP).

V

VAST
See volume access structure table.

virtual disk
In the Data Transfer System (DTS), a feature that permits a file on the host system to be assigned as a personal computer (PC) disk drive. Once assigned, a virtual disk appears to the PC user to be a disk drive that is physically attached to the PC.

virtual memory
A system technique that treats disk storage space as an extension of main memory, giving the appearance of a larger main memory than actually exists.

virtual static imaging device (VSID)
A device that produces high-speed, high-quality printouts. A laser printer is an example of a VSID.

volume
The medium of a mass storage device such as a disk, disk pack, or tape reel. The term *volume* is not restricted to the volume library on a cataloging system or the volume directory on a system with tape volume security. For example, on the BTOS family of workstations, the hard disk is a volume, and each floppy disk is a volume. When a volume is initialized, it is assigned a volume name and an optional password.

volume access structure table (VAST)
A section of the catalog file on a cataloging system that the system uses to access the volume library.

volume directory
A section of the catalog that tracks the status of tapes on a system that uses the tape security subsystem.

volume label
The area on a disk or tape that stores the name and serial number assigned to the volume.

volume library
On a cataloging system, a section of the catalog that keeps track of various tape and disk volumes by serial number. The Work Flow Language (WFL) *VOLUME ADD* statement places the initial entry for a given tape or disk into the volume library. Tapes or disks listed in the volume library are said to be "volumed."

volumed disk
On a cataloging system, a disk that has been entered into the volume library with the Work Flow Language (WFL) *VOLUME ADD* statement so that the disk can be used to store cataloged files.

VSID
See virtual static imaging device.

W

wait queue
Any of a number of lists of processes, maintained by the operating system, that are waiting on internal events. For example, a process that is waiting for an I/O to complete is placed in a wait queue.

WFL
See Work Flow Language.

WFL job
(1) A Work Flow Language (WFL) program, or the execution of such a program. (2) A collection of Work Flow Language (WFL) statements that enable the user to run programs or tasks.

window
In the Communications Management System (COMS) architecture, the concept that enables a number of program environments to be operated independently and simultaneously at one station. One of the program environments can be viewed while the others continue to operate.

WORM
Write-once, read many. *See also* WORM device.

WORM device
A random-access optical disk drive that can write to the WORM media only once, but can access the files on the WORM media many times.

WORM medium
A high-density storage medium that can be written to only once and can be read from many times.

Work Flow Language (WFL)
A Unisys language used for constructing jobs that compile or run programs on A Series systems. WFL includes variables, expressions, and flow-of-control statements that offer the programmer a wide range of capabilities with regard to task control.

Bibliography

A Series A 1 - A 6 Systems Software Installation Guide (form 8600 1468). Unisys Corporation. Formerly *A Series A 1/A 4 Software Installation Guide,* the *A Series System Software Installation Guide, Volume 1: A 2/A 3,* the *A Series System Software Installation Guide, Volume 1: A 3 Advanced,* the *A Series System Software Installation Guide, Volume 1: A 3 Dual-Processor,* the *A Series A 4 Model FX Software Installation Guide,* and the *A Series A 5/A 6 Software Installation Guide.*

A Series CANDE Configuration Reference Manual (form 8600 1344). Unisys Corporation.

A Series CANDE Operations Reference Manual (form 8600 1500). Unisys Corporation.

A Series Communications Management System (COMS) Capabilities Manual (form 8600 0627). Unisys Corporation.

A Series Communications Management System (COMS) Configuration Guide (form 8600 0312). Unisys Corporation.

A Series Communications Management System (COMS) Operations Guide (form 8600 0833). Unisys Corporation.

A Series Communications Management System (COMS) Programming Guide (form 8600 0650). Unisys Corporation.

A Series Data Communications Protocols Installation and Implementation Guide (form 8600 0486). Unisys Corporation.

A Series Disk Subsystem Administration and Operations Guide (form 8600 0668). Unisys Corporation.

A Series DMSII Utilities Operations Guide (form 8600 0759). Unisys Corporation.

A Series Documentation Library Overview (form 8600 0361). Unisys Corporation.

A Series File Attributes Programming Reference Manual (form 8600 0064). Unisys Corporation. Formerly *A Series I/O Subsystem Programming Reference Manual.*

A Series GETSTATUS/SETSTATUS Programming Reference Manual (form 8600 0346). Unisys Corporation.

A Series I/O Subsystem Programming Guide (form 8600 0056). Unisys Corporation. Formerly *A Series I/O Subsystem Programming Reference Manual.*

A Series Integrated Communications Processor (ICP1) Memory Dump Analyzer Operations Reference Manual (form 1182383). Unisys Corporation.

A Series Interactive Datacomm Configurator (IDC) Operations Guide (form 1169810). Unisys Corporation.

A Series Mail System Installation and Administration Guide (form 8600 1559). Unisys Corporation.

A Series Mail System Operations Guide (form 8600 0427). Unisys Corporation.

A Series Memory Subsystem Administration and Operations Guide (form 1169836). Unisys Corporation.

A Series Menu-Assisted Resource Control (MARC) Operations Guide (form 8600 0403). Unisys Corporation.

A Series MultiLingual System (MLS) Administration, Operations, and Programming Guide (form 8600 0288). Unisys Corporation.

A Series Network Definition Language II (NDLII) Programming Reference Manual (form 1169604). Unisys Corporation.

A Series Operating System Installation Guide (form 8600 1021). Unisys Corporation.

A Series Physical I/O Technical Overview (form 1169943). Unisys Corporation.

A Series Print System (PrintS/ReprintS) Administration, Operations, and Programming Guide (form 8600 1039). Unisys Corporation.

A Series Printing Utilities Operations Guide (form 8600 0692). Unisys Corporation.

A Series Security Administration Guide (form 8600 0973). Unisys Corporation.

A Series Security Features Operations and Programming Guide (form 8600 0528). Unisys Corporation.

A Series Software Disk Cache Module Installation and Operating Guide (form 3950 8874). Unisys Corporation.

A Series Software Release Installation Guide (form 8600 0981). Unisys Corporation.

A Series System Architecture Reference Manual, Volume 2 (form 5014954). Unisys Corporation.

A Series System Commands Operations Reference Manual (form 8600 0395). Unisys Corporation.

A Series System Configuration Guide (form 8600 0445). Unisys Corporation.

A Series System Management Facility II (SMFII) Query Operations Guide (form 7831 1867). Unisys Corporation.

A Series System Management Facility II (SMFII) Resource Management Operations Reference Manual (form 7831 1628). Unisys Corporation.

A Series System Operations Guide (form 8600 0387). Unisys Corporation. *A Series System Software Installation Guide, Volume 1: A 9* (form 1169695). Unisys Corporation.

A Series System Software Installation Guide, Volume 1: A 10 (form 1169935). Unisys Corporation.

A Series System Software Support Reference Manual (form 8600 0478). Unisys Corporation.

A Series System Software Utilities Operations Reference Manual (form 8600 0460). Unisys Corporation.

A Series Systems Functional Overview (form 8600 0353). Unisys Corporation.

A Series SYSTEMSTATUS Programming Reference Manual (form 8600 0452). Unisys Corporation.

A Series Task Attributes Programming Reference Manual (form 8600 0502). Unisys Corporation.

A Series Task Management Programming Guide (form 8600 0494). Unisys Corporation.

A Series Work Flow Language (WFL) Programming Reference Manual (form 8600 1047). Unisys Corporation.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

UNISYS

Help Us To Help You

Publication Title

Form Number

Unisys Corporation is interested in your comments and suggestions reguarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments:

Name Telephone number ( )

Title Company

Address

City State Zip code

Cut along dotted line

Tape

Please Do Not Staple

Tape

Fold Here

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 817 DETROIT, MI

POSTAGE WILL BE PAID BY ADDRESSEE

UNISYS CORPORATION
ATTN: PUBLICATIONS
25725 JERONIMO ROAD
MISSION VIEJO, CA 92691-9826

86000437-000