Multics

System Administration Procedures

Honeywell Bull

MULTICS

SYSTEM ADMINISTRATION PROCEDURES

SUBJECT

System Administration Procedures for the Multics System, Including Those Details Required for Controlling Resources, Managing the Storage System, System Security, I/O Daemons, Projects and Users, System Usage Control, Accounting Operations, and Other System Options

SPECIAL INSTRUCTIONS

This document supersedes the Multics Administrator's Manual – System, Order No. AK50-02, dated July 1982, and its associated addenda, Addendum A, dated February 1983, and Addendum B, dated December 1983. Because this manual has been completely rewritten, change indicators have not been used. The Multics Administrator's manual set has been reorganized. Refer to the "Significant Changes" section of the Preface for additional information.

SOFTWARE SUPPORTED

Multics Software Release 11.0

ORDER NUMBER

AK50-03

May 1985

Honeywell Bull

PREFACE

This document describes procedures intended to be performed by individuals filling the role of system administrator. System administrators provide their sites with a particular Multics operating environment. They are responsible for such tasks as controlling and allocating resources, registering projects and users, creating load control groups, setting prices on resources, setting limits on and billing for resource usage, scheduling system activities such as hours of operation, shift change times, and unattended service, describing site parameters and setting site options, and assuring system security.

This document makes reference, as needed, to the various commands required to implement the specified procedures. A detailed description of all the commands mentioned in this manual can be found either in the *Multics Commands and Active Functions* manual (Order No. AG92) or the *Multics Administration, Maintenance, and Operations Commands* manual (Order No. GB64). The Commands and Active Functions manual describes commands available for general use. The Administration, Maintenance, and Operations Commands manual describes those commands available only to "privileged" users.

Other administrative documents available include the *Project Administrator's Guide* (Order No. AK51) and the *Multics Communications Administration* manual (Order No. CC75). The Project Administrator's Guide contains a subset of the information described in this manual and is intended for those individuals who are delegated the duties of project administrator. The Communications Administration manual contains information specific to site communications administration.

Significant Changes in AK50-03B

A description of the Mail Table entry has been added to Section 4.

Section 18 includes a paragraph on access to the set_proc_required.acs segment.

The History Log File has been explained in greater detail in Section 24.

In Section 25 a description of absentee process manipulation and message formats describing nonstandard audit messages requesting to set the process_termination_monitor, to get com_channel_info and notifications of PNT changes have been added.

Section 33 includes a description of the Mail Table and how to modify it.

CONTENTS

Illustrations

Tables

PART I
INTRODUCTION

SECTION 1

INTRODUCTION

HOW TO USE THIS MANUAL

This manual is divided into 11 parts: each part is designed to describe and explain a major aspect of system administration. The parts contained in this manual are:

- Introduction
- System Description
- Controlling Resources
- Managing the Storage System Hierarchy
- Managing Storage System Volumes
- Assuring System Security
- Managing I/O Daemons
- Managing Projects and Users
- Controlling System Usage
- Managing the Accounting Subsystem
- Setting Other System Options

The Introduction (Section 1 and Section 2) includes an overview of Multics system administration and the responsibilities of a Multics system administrator.

The System Description (Section 3 through Section 5) is an overview of system hardware and system tables and data bases, and contains a glossary of system terms.

Controlling Resources (Section 6 through Section 11) describes resource management with the Resource Control Program, and the procedures required for setting rates for resource usage.

Managing the Storage System Hierarchy (Section 12 and Section 13) provides a description of all system directories and the procedures for setting quota for projects and directories.

Managing Storage System Volumes (Section 14 through Section 17) describes disk volume management, logical volume management, and the procedures for administering a quota account.

Assuring System Security (Section 18 through Section 26) describes the procedures necessary to assure the security of the file system, RCPRM, communications channels, I/O daemons, the absentee facility, and privileged operations. It also describes system logs, security auditing of system logs, and miscellaneous security tasks.

Managing I/O Daemons (Section 27 through Section 31) describes I/O daemon registration and administration.

Managing Projects and Users (Section 32 and Section 33) describes the registration procedures for projects, users, and special user identities.

Controlling System Usage (Section 34 through Section 37) describes the procedures for managing shifts and absentee usage and allocating processor resources.

Managing the Accounting Subsystem (Section 38 and Section 39) describes accounting and billing procedures.

Setting Other System Options (section 40 through Section 44) describes the management of settable system parameters with the ed_installation_parms command, delegating responsibilities among administrators, modifying the system time tables, managing the system mail table, and data management administration procedures.

Command descriptions are not included in this manual. For details on administrative commands, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

GETTING HELP WITH COMMANDS

Info segments for the following types of commands can be found in the following directories:

Commands	Directories
user	>doc>info
privileged user	>doc>privileged
ring_4 initializer	>doc>ss>operator
ring_1 initializer	>doc>ss>r1_initializer
accounting	>doc>ss>accounting
I/O daemon	>doc>ss>io_daemon
BCE	>doc>ss>bce
BOS	>doc>ss>bos

Access to the above info segments depends upon your current environment.

If you are in the user command environment, you can view info segments for user commands by using the user "help" command. For example:

```
help list
```

You can view info segments for all other commands by using the user "help" command and specifying the absolute pathname of the info segment. For example:

```
help >doc>privileged>set_system_console
```

Or you can add one or more directories to your info search paths by adding a line to your start_up.ec. For example:

```
asp info >doc>privileged>doc>ss>rl_initializer
```

Then you can view info segments in the directories in the info search list without specifying the absolute pathname.

If you are in the ring_4 initializer command environment, you can view info segments for ring_4 initializer commands by typing the initializer "help" command. For example:

```
help down
```

If the info segment you want to look at is for an exec command, specify its name in the format "x.<command_name>. For example:

```
help x.attend
```

If you are in any of the following environments, you cannot look at info segments at all:

- ring_1 initializer
- restricted accounting
- I/O daemon
- BCE
- BOS

For a more detailed discussion of command environments, refer to the *Administration, Maintenance and Operations Commands* manual, Order No. GB64.

SECTION 2

SYSTEM ADMINISTRATION OVERVIEW

MULTICS SYSTEM ADMINISTRATION

The Multics system administrator is responsible for providing an overall operating environment for projects and users on the system. Multics administration can be divided into four administrative areas:

- System administration
- Project administration
- Accounting administration
- Security administration

The only administrator that a Multics site is required to have is a system administrator. The system administrator may choose to delegate the responsibilities in the areas of project, accounting, or security management to trusted individuals. When other types of administrators are used at a site, the system administrator must still know how to function in all administrator roles. For example, an accounting administrator may experience problems with billing programs that only a system administrator can solve. Also, all projects are initially under the control of a system or accounting administrator until control is delegated to a project administrator.

An individual becomes a system administrator by being registered on the SysAdmin project. SysAdmin processes are, by default, given access to major directories and potential access to all segments and directories in the hierarchy. This allows the system administrator access to all aspects of the system for the purpose of managing system projects and users, and controlling access to system resources.

As described in the previous section, Multics administrative responsibilities include the management of:

- resources
- storage system hierarchy
- storage system volumes
- security
- I/O daemons
- projects and users
- system usage
- accounting subsystem
- miscellaneous system options

Resource and System Usage Management

Multics resource administration tools let you control the usage of and charges for the following on a per-project basis:

- interactive CPU time
- interactive real time
- absentee CPU time and memory units
- I/O daemon usage
- disk storage
- communications channel usage

Each of the above is described in detail in Section 6. Multics Resource Management (RCPRM) software is also under the control of the system administrator and is described in Part III of the manual.

In addition, the system administrator can determine:

- system load units
- event scheduling
- system workloads

Each load control group has a quota of primary load units that represent the number of users within the group who will always be able to log in to the system. Users who exceed the primary quota are assigned "secondary status" and can log in to the system if it is not full. Users with secondary status can be preempted by users with "primary status" if the system is full. Load units can be set for projects and users depending upon their processing requirements. Load unit allocation is described in Part IX, "Controlling System Usage".

Event scheduling lets you establish user and job priorities by flexibly managing batch usage (absentee usage) and interactive usage. Management of absentee usage is also described in Part IX, "Controlling System usage".

System workloads can be implemented by grouping specific users into work classes and allocating each class a guaranteed percentage of CPU resources. This permits each class of users to perform their data processing functions regardless of the total system workload. A work class can be an entire load control group, a single user, or a class of users like all I/O daemons or all absentee jobs in a given queue. Membership in a specific class and the percentage of CPU time assigned can be changed at any time.

Storage System Management

Storage system management involves the management of the storage system hierarchy and the management of the storage system volumes. The storage system hierarchy includes all administrative related directories. It is your responsibility as system administrator to be thoroughly familiar with the contents of all administrative directories, set quota for administrative directories and nonadministrative directories, and control access to critical administrative directories. See Part IV, "Managing the Storage System Hierarchy" for detailed information on administrative directories and quota management. See Section 18 for a complete description of access requirements for administrative directories.

Storage system volume management involves disk volume registration, logical volume access management, logical volume use, and quota account use. Storage system volume management is described in Part V.

Security Management

As system administrator, you are responsible for the secure operation of the system. This includes maintaining the security of:

- the file system
- RCPRM
- Communications channels
- I/O daemons
- the absentee facility
- privileged operations
- system logs and security auditing
- device and logical volume access
- Other security tasks.

See Part VI, "Assuring System Security", for complete description of system security management. Device access and logical volume access are described in Section 7 and Section 15, respectively.

I/O Daemon Management

The I/O daemon handles all printing, punching, and card input. As system administrator, you are responsible for setting up the I/O daemon, managing I/O daemon input and output, and running the I/O daemons. Part VII, "Managing I/O Daemons", contains a detailed description of I/O daemon administration.

Project and User Management

The registration of all new projects and new users on the system is the responsibility of the system administrator. Administration of projects and users can be decentralized by delegating a project a project administrator at registration time. This means that you, as system administrator, can allocate resources to specific projects and produce individual accounting reports for each project. If a project is delegated a project administrator (see Section 32), the project administrator can allocate resources to users of a specific project on an individual basis.

Part VIII, "Managing Projects and Users" contains a complete description of project registration and user registration.

Accounting Subsystem Management

As system administrator you can establish up to eight separate work shifts with different rates applied to each shift, establish automatic accounting procedures, and produce accounting reports and bills. You can also choose to delegate all accounting administration to an accounting administrator. Multics accounting software permits project administrators to do their own sub-billing. A complete description of accounting management is provided in Part X, "Managing the Accounting Subsystem".

Miscellaneous Management Functions

Other administrative functions include regulation of the settable parameters in installation_parms with the ed_installation_parms command. See Part XI, "Setting Other System Options" for a description of the settable parameters in installation_parms.

PART II
SYSTEM DESCRIPTION

SECTION 3

HARDWARE OVERVIEW

A computer system consists of *hardware, software* and *firmware*. Hardware refers to all of the physical devices and electronic circuitry. Software refers to all of the *programs* that control the activities of the computer. A program is a set of instructions for solving a problem, coded in a language the computer understands. The software is said to *run* or *execute* on the hardware. Firmware is specialized software that is attached to hardware but subordinate to it. An overview of Multics hardware is presented in this section.

MAJOR MODULES

The major hardware modules described in this section are often referred to as *boxes* or *units*.

Memory and System Controller Unit (SCU)

Memory is the part of a Multics configuration that contains instructions and manipulated data. *SCUs* control memory. They also control and coordinate the activities of all other hardware modules. Each SCU has a specific amount of memory associated with it, divided into parts known as *store units*. SCUs interface between memory and CPUs and IOMs. (The CPU and the IOM are described next in this section.) SCUs also contain facilities which allow CPUs and IOMs to communicate with each other. In addition, SCUs contain calendar clocks. There are two kinds of SCUs: the Level 68 and the DPS 8. The primary difference between them is that on the DPS 8 SCU, certain switches and lights have been replaced with terminal displays. (See "DPS 8 vs Level 68" later in this section.) There can be up to eight SCUs in a Level 68 configuration, and up to four SCUs in a DPS 8 configuration. An SCU is often called a *system controller*.

Central Processing Unit (CPU)

CPUs are responsible for performing most of the computational processing done by the system. This means that they perform most of the arithmetic and logical manipulations of data. There are two kinds of CPUs: the Level 68 and the DPS 8. The primary difference between them is that on the DPS 8 CPU, certain switches and lights have been replaced with terminal displays. (See "DPS 8 vs Level 68" later in this section.) In addition, DPS 8 submodels have different performance rates from the Level 68, some faster, some slower. There can be up to 7 CPUs in a Multics configuration. A CPU is often called a *processor*.

Input/Output Multiplexer (IOM)

IOMs manage all of the *peripherals* connected to the system. Peripherals include the bootload console, terminals, storage devices, unit record devices, FNPs, and miscellaneous other devices. Storage devices and unit record devices are connected to MPCs (described later in this section), which are in turn connected to IOMs. Managing the peripherals means that IOMs handle all transfer of data between them and memory. To help with the transfer of data, IOMs use *IOM channels*. An IOM channel is a connection between an IOM and an FNP, an MPC, or a console, over which the system can do I/O. The configuration deck specifies what channels exist and to what they are connected. There are two kinds of IOMs: the Level 68 and the DPS 8. The primary difference between them is that on the DPS 8 IOM, certain switches and lights have been replaced with terminal displays. (See "DPS 8 vs Level 68" later in this section.) There can be up to 4 IOMs in a Multics configuration.

Front-End Network Processor (FNP)

FNPs are responsible for data communications. This means that they provide the physical and logical connections between the system and terminals, *networks*, and other computers. A network is a collection of hardware that provides the service of connecting many different pieces of data processing equipment, allowing them to communicate with each other. FNPs provide the physical connections by using *communications lines*. A communication line can be anything from a simple pair of wires in a circuit to the entire telephone system. One important use of communications lines is to connect IOMs with devices (like offsite printers) which are far away, since IOMs are designed to communicate only with devices which are nearby. FNPs are nearby, so devices which are far away are connected to FNPs, and communicate with IOMs via FNPs. FNPs provide the logical connections by using *communications channels*. A communications channel defines connections. It is read from or written to. There can be up to eight FNPs in a Multics configuration. An FNP is often called a *multiplexer*, a *front-end*, a *communications processor*, or a *datanet*. "Datanet" comes from the model name of FNPs--DATANET 6670. Although an FNP is a major module, it is often referred to as one of the peripherals.

Microprogrammed Peripheral Controller (MPC)

MPCs control either storage devices (disk drives and tape drives) or unit record devices (printers, card punches and card readers). Some MPCs are *cross barred*. This means that they are connected to more than one IOM. The advantage of this is that if one of the IOMs breaks, the MPCs connected to it can still run, because the other IOM will pick up its load. An MPC's connection to an IOM is called a *link adapter*. An MPC can have one or two link adapters. If it has two, they are usually connected to different IOMs.

For an illustration of cross barred MPCs, see Figure 3-1.

PERIPHERALS

Peripherals are devices which are connected to a Multics system configuration and controlled by it. Peripherals include terminals, storage devices (tape drives and disk drives), unit record devices (printers, card punches and card readers), FNPs (described earlier), the bootload console, and miscellaneous other devices. They communicate with the CPU through the IOM.

Terminals

Terminals are devices used to send input to the system and receive output from the system. There are basically two types of terminals: *hardcopy* or *printing terminals*, and *video* or *crt* (cathode ray tubes) *terminals*. Both have keyboards that resemble those on typewriters. A hardcopy terminal prints input and output on paper. A video terminal displays input and output on a television-like screen.

Storage Devices

DISKS

Disks are the principal means of storing information on Multics. An individual disk unit is called a *disk pack*. The device which houses packs is called a *disk drive*. Some disk drives are *cross barred*. This means that they are connected to more than one MPC. The advantage of this is that if one MPC breaks, the disk drives connected to it can still run. because the other MPC will pick up its load.

For an illustration of cross barred disks, see Figure 3-1.

Multics uses disks in two ways: as *user I/O disks* and as *storage system disks*. A user I/O disk belongs to a user. It can contain any kind of data in any format. A storage system disk belongs to the system. It contains some part of the storage system hierarchy (described in Section 3), in a standard format. One important quality of a disk pack is its capacity for shared access. Information stored on a disk pack can be used by many people at the same time (as long as they've been given proper access). However, the owner of a user I/O disk pack may be the only person authorized to use it.

You will often hear the word "volume" used in reference to disks. A *physical volume* is a set of data accessed as a group. Some disk packs contain one physical volume, while others contain two. A *logical volume* is a set of physical volumes which have some logical relationship to each other.

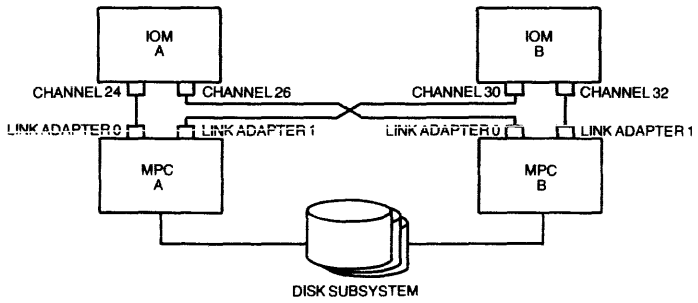

Figure 3-1. Cross Barred MPCs and Disks

There are different models of disks. The first way they differ is in whether they contain one physical volume or two. The second way they differ is in whether or not the packs are demountable. The third way they differ is in how much information they can hold. Older disks, such as the 451s, contain one physical volume, have demountable packs and hold less information. Newer disks, such as the 500s and 501s, contain two physical volumes, have packs which are not demountable, and hold more information.

MASS STORAGE PROCESSOR (MSP)

MSPs are the MPCs which control disk drives. They are often called *disk controllers.*

TAPES

Tapes are another means of storing information on Multics. It's cheaper to store data on a tape than on a disk, but it takes longer to get data on and off a tape. An individual tape unit is called a *tape reel*. The device which houses reels is called a *tape drive* or a *tape handler*. Some tape drives are *cross barred*. This means that they are connected to more than one MPC. The advantage of this is that if one MPC breaks, the tape drives connected to it can still run, because the other MPC will pick up its load.

You may hear the word "volume" used in reference to tapes. A tape volume is a tape reel.

Differing amounts of information can be stored on tapes, depending on how they are recorded. A number which expresses the amount of information stored on a tape is called the tape's *density*. A tape with a density of 6250 bpi (bits-per-inch) holds much more information than a tape with a density of 1600 bpi.

MAGNETIC TAPE PROCESSOR (MTP)

MTPs are the MPCs which control tape drives. They are often called *tape controllers.*

Unit Record Devices

PRINTERS

Printers are strictly output devices. They do not accept input. Printers provide hardcopy (paper) output in response to software programs. Sometimes they are called *line printers* because they print out information one line at a time. They are also called *high-speed printers*, because they print information at very high speeds, much faster than terminals.

CARD PUNCHES

Like printers, *card punches* are strictly output devices. They do not accept input. Card punches provide punched card output in response to software programs.

CARD READERS

Card readers transfer programs and data punched on computer cards to the central system. A necessary accompanying device to a card reader is a *keypunch machine*, a typewriter-like device with which you can type characters onto and punch holes into computer cards. There is also a device which functions as both a card reader *and* a card punch. This device is known as a *Combined Card Unit* (CCU). The procedures for reading and punching cards with a CCU are the same as those used with separate readers and punches.

UNIT RECORD PROCESSOR (URP)

URPs are the MPCs which control printers, card punches and card readers. They are often called *unit record controllers*.

PANELS

All major hardware modules have at least one collection of switches and lights on them, called a *panel*. Panels are used to control the hardware and to make changes in the way the hardware operates.

For an illustration of a Multics system configuration, see Figure 3-2.

DPS 8 VS LEVEL 68

There are two kinds of Multics systems: the original *Level 68* system and the new, improved *DPS 8* system. (DPS stands for *Distributed Processing System.*) Internally, the DPS 8 processor works differently from the Level 68 processor, but architecturally, they support the same set of instructions and registers. For users, the primary difference between the two processors is that some of the DPS 8 submodels are faster. For you, the primary difference is that DPS 8 processors, as well as SCUs and IOMs, do not have maintenance panels. The information that is provided by these panels in a Level 68 system is provided by displays on a terminal in a DPS 8 system. The exact panels which do not exist on DPS 8 boxes are as follows:

```
CPU:  Maintenance, Test and Display Panels
SCU:  Maintenance and Display Panels
IOM:  Maintenance and Test Panels
```

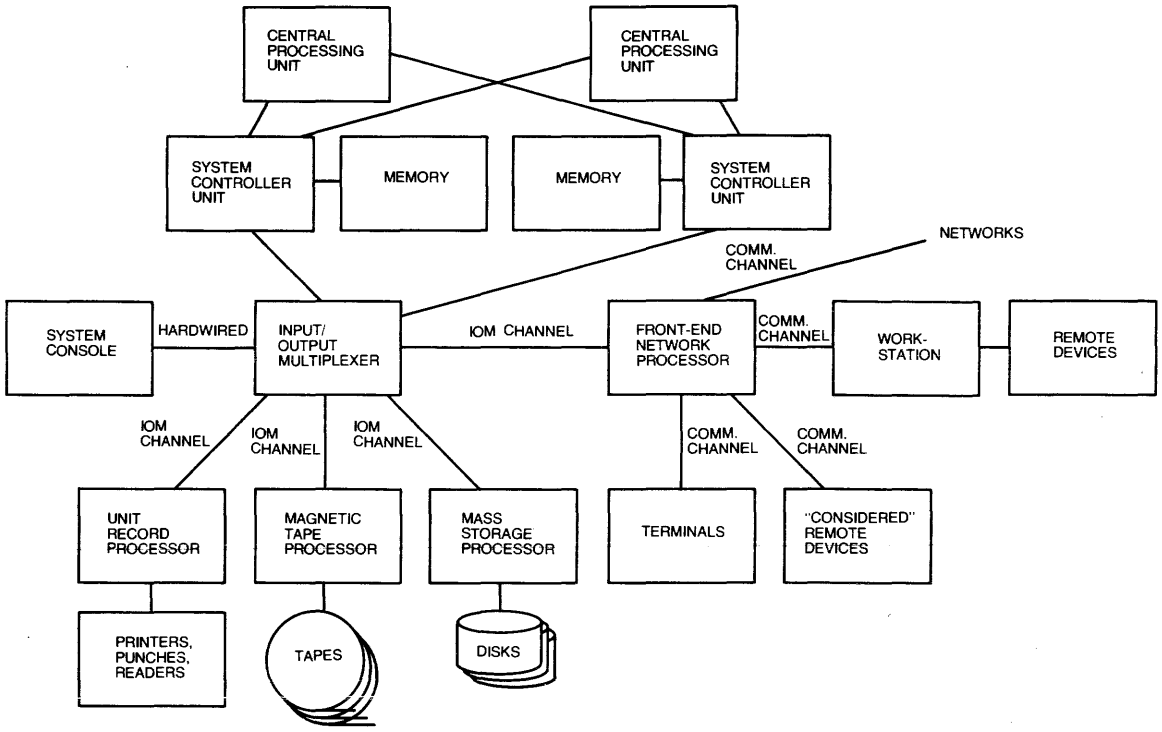

Figure 3-2. A Multics System Configuration

The displays which replace the maintenance panels are produced by the *Dynamic Maintenance Panel* (DMP), which is part of the processor. Displays from the DMP may be accessed in either of two ways: with a standard VIP terminal attached to the DMP or with a *Diagnostics Processor Unit* (DPU). The DPU serves as an interface to the DMP for the processor, the SCU and the IOM. Your decision as to whether you should use the DPU or the VIP attached to the DMP will depend on the configuration at your site. You might have one VIP and a patching mechanism to connect it to the desired DMP interface, or a separate terminal for each DMP interface, and some combination of these. You might or might not have a DPU. If you have questions about this decision, you should consult with your system administrator.

The DPU is a stand alone computer system, which has a VIP terminal attached to it (currently a VIP7205). You should not confuse this terminal, which is part of the DPU subsystem, with the terminal mentioned above, which is connected directly to the DMP.

The DPU provides a maintenance capability that includes remote maintenance control of the DPS 8 processor, SCU and IOM. In other words, the switch settings and the contents of various registers for these units may be displayed on the attached terminal.

In addition to the above-mentioned panels being replaced, the configuration panels on these three units are packaged somewhat differently than they are on the Level 68 machines. The FNP and memory have not changed with DPS 8.

For normal operations, you will find little difference between the procedures you follow for the DPS 8 and those you follow for the Level 68. However, there is a difference in some procedures for recovering from system failures. The most notable of these is executing switches to return to BCE.

SECTION 4

OVERVIEW OF SYSTEM TABLES/DATA BASES

This section presents an overview of the function and contents of system tables and data bases. The procedure for removing old versions of system tables with the date_deleter command is given below.

CONTENTS AND FUNCTION OF SYSTEM TABLES AND DATA BASES

System Administrator Table (>sc1>sat)

The System Administrator Table (SAT) is a binary table specifying the projects that use the system, the privileges assigned to these projects, and the project administrators, if any. The accounting administrator and the system administrator both use the edit_proj command to maintain the SAT. Additionally, the system administrator can modify the contents of all SAT entries using the entry point, edit_proj$change_all.

The SAT contains a header and project entries, one for each project registered on the system. The SAT header includes the following variables:

- The Person_id.Project_id of one or two system administrators. An asterisk (*) is permitted in either component.
- The user weight table (UWT). This table lists process overseer procedures and an associated weight in load units. (If a process overseer not listed in the SAT is used, a default weight is assigned.)

To change these header variables, use the admin_util command. For a complete description of this command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Master Group Table (>sc1>mgt)

The Master Group Table (MGT) is a segment containing information about the load control group and work classes used at the site. The work class and load control group membership of each user is determined from information in the System Administrator Table (SAT) and the Project Master File (PMF) (see "Managing Projects" later in this manual). The system administrator maintains and edits the MGT using the ed_mgt command which is described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Project Master File (>udd>Proj_dir>Proj_id.pmf)

The Project Master File (PMF) is an ASCII segment giving the names, attributes, and account limits of the users of a particular project. It is created and modified using any Multics editor and is converted into a project definition table (PDT) using the cv_pmf command which is described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Project Definition Table (>udd>Proj_dir>Proj_id.pdt)

The Project Definition Table (PDT) is a binary table defining the membership of a project and the attributes of the project's users. The PDT is referenced when a user logs in. This table is the binary version of the project master file (PMF), and it contains accounting information for each user on the project.

Person Name Table (>sc1>PNT)

The Person Name Table (PNT) is a multisegment file (msf) referenced by the system when a user attempts to log in or use the card input facility; it contains all valid Person_ids. Associated with each Person_id in the table is a login alias, default project, password, network password, date and time of last login, and terminal identification of last login. (Refer to "Managing Users" later in this manual).

Mail Table (MT) (>site>mail_system_dir>MAIL_TABLE)

All users are registered in the MAIL TABLE (MT). Basically, the mail table allows mail to be addressed to another user without specifying that user's project and merely knowing his Person_id or an alias. For example, you can send mail to the user Willow.ProjCat using only the identifier "Willow". Additionally, if user Willow is registered in the mail table with an alias of "Jim," you can send mail using only that identifier. Likewise, other users can send mail or messages to you by name alone.

The mail table also allows mail to be sent to mailing lists and Forum meetings. Such "non-users" can have their own system-wide names entered in the mail table.

See "Understanding the Mail Table" in Section 33.

User Registration File (>udd>sa>a>URF)

The User Registration File (URF) is a multisegment file containing each user's full name, mailing address, programmer number, and Person_id. This data base is referenced when registering a new user and when performing administrative operations. To print the contents of the URF use the print_urf command listed in the *Multics Administration Maintenance, and Operations Commands* manual Order No. GB64.

Resource Type Master File (>udd>sa>a>RTMF)

The Resource Type Master File (RTMF) is an ASCII segment converted by the cv_rtmf command to produce the Resource Type Description Table (RTDT). The RTMF describes all resource types on the system. It is an ASCII file, which when compiled into a binary table, is installed by the answering service at the system administrator's request. The RTMF consists of a series of entries, one for each resource type that is known to the system. Each entry consists of a series of statements and substatements of the form: <keyword>: <parameter> .

The RTMF consists of one or more complete resource type entries. Each entry must include the name of the resource type being described, fixed parameters for that resource type, and default parameters to be applied at registration time in the absence of explicit parameters. In addition, a resource type entry may contain one or more sets of special registration parameters. See "Managing the Resource Type Master File" (RTMF) in this manual.

Resource Type Description Table (>udd>sa>a>RTMF.rtdt)

The Resource Type Description Table (RTDT) is a binary table containing an entry for each resource type (e.g., tape_drive, tape_volume) in the system. It cannot be examined or modified with text the editors. The RTDT describes all of the resource types known to the system--both device types and volume types. For each resource type, it specifies the attributes that a resource of this type may possess and the type of resource(s) that may be used with it. For example, an RTDT would say that tape_drive and tape_volume are resource types known to the system. It would also indicate that a tape_drive is a device and that a tape_volume is a volume. It might say that this type of device can be either 7 or 9 track and can be used at densities of 800 or 1600. It would also mention that a tape_drive is used in conjunction with a tape_volume.

Channel Master File (>udd>sa>a>CMF)

The Channel Master File (CMF) is an ASCII file containing descriptions of all terminal channels on the system. It is converted by the cv_cmf command to create the CMF.cdt segment. It is installed at the system administrator's request. It describes the protocol, modems, etc. that each channel uses. Access checks and AIM attributes are also definable for each channel.

Channel Definition Table (>udd>sa>a>CMF.cdt)

The channel definition table (cdt) is a binary version of the CMF. The cdt is installed in >sc1 using the install command.

Terminal Type File (>udd>sa>a>TTF)

The Terminal Type File (TTF) is an ASCII file which, when compiled into a binary table (the TTT), is installed at the system administrator's request. The TTF consists of a series of entries describing terminal types, tables, and answerback interpretations. Each entry consists of a series of statements that begin with a keyword and end with a semicolon. The terminal type describes the characteristics of the remote system.

Terminal Type Table (>udd>sa>a>TTF.ttt)

A data base that resides by default in the segment >system_control_1>ttt and describes all the terminal types used by MCS. The TTT is a binary table containing numbers and pointers as well as character strings; therefore, it cannot be examined or modified using the editors. The display_ttt command prints out all or part of the TTT; when the system administrator wishes to add or delete terminal types, or change the information about one or more terminal types, he compiles a TTF into a TTT using the cv_ttf command, and then uses the install command to replace the copy of the TTT in the system. The cv_ttf command is described in the *Multics Commands and Active Functions* manual, Order No. AG92. The install command is described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

I/O Daemon Tables (>udd>idd>iod_tables)

The iod_tables is an administrative date base that can be adapted to the specific needs of a particular Multics site. This data base contains several different tables of information and is referred to as the "I/O daemon tables." I/O Daemon Tables are compiled with the iod_tables_compiler.

billing segments (>udd>sa>a)

The billing segments are contained in the >udd>sa>a directory. This directory is the working directory for the accounting administrators and contains all of the local site accounting data bases. The billing segments contained in the >udd>sa>a directory are listed below.

billing_footnote
Segment of announcements (e.g., forthcoming price changes) to project supervisors that is printed at the bottom of each project's usage summary report (on the mailing copy only); this segment is optional.

disk_stat
A binary segment that gives disk quota and usage for all directories with quota; it is prepared daily.

miscfile
The miscellaneous charges and credits journal.

PDTs (safe_pdts)
A directory that contains the SAT and the PDTs. The crank copies the SAT and the PDT from the >sc1 directory into the safe_pdts directory. These copies are used daily, to produce month-to-date charge summaries, and, at the end of a billing period, to produce the bills and to reset the usage charges in PDTs stored in >sc1>pdt.

projfile
The project registration segment, containing descriptive information about each project, including title, supervisor address, and disk storage usage.

reqfile
The requisition segment, containing account number, billing address, total charges, and requisition number for each usage account.

today.use_totals
A statistical data base that describes month-to-date system resource availability and usage; prepared daily from meter_data and the PDTs.

REMOVING OLD VERSIONS OF SYSTEM TABLES AND OLD I/O SEGMENTS

The date_deleter command performs a delete-by-date operation in a directory by removing all segments and multisegment files older than a specified number of days. The date_deleter command should be used carefully, since it will delete segments that are used all the time, but have not been modified recently.

In the date_deleter command, "n_days" is the number of days that must have elapsed since a segment was last modified in order for it to qualify for deletion. The time elapsed is measured from date_time_contents_modified. The "star_names" are the optional starnames of files to be deleted. If none are supplied, all files older than the designated number of days are deleted; otherwise, only files matching one or more of the starnames, and older than the specified number of days, are deleted.

To use the date_deleter command, type:

```
date_deleter >udd>Proj>listing_pool 2**.list
```

This command line deletes all listing files in the >udd>porj>listing_pool directory that are more than two days old.

The following example explains how to delete old versions of the CMF. Old versions of the CMF are left in >scl>comms with names CMF.1, CMF.2, etc. Old versions of CMF.cdt are also left there with names CMF.cdt.1, CMF.cdt.2, etc. Additionally, these segments have "shriek names" beginning with a "", such as BBBJKQDJbFDdLw. Old "io" segments (used by update_seg) are left in >scl>comms with names CMF.date.time.io. These segments may be deleted after some appropriate period of time, such as a week or a month. The date_deleter command can be used to do this. For example, to delete all such segments that are older than 30 days:

```
! date_deleter >scl>comms 30 CMF.* CMF.cdt.* CMF.*.*.io
```

SECTION 5

GLOSSARY OF SYSTEM TERMS

The following list contains brief descriptions of important system terms.

access control segment (ACS)
A zero-length segment associated with some system resource or resources; the effective access mode of a user process to an ACS determines the access of that process to the resource identified with it. The entryname of an ACS generally describes the system resource it is associated with; for example, the ACS for tape drive 6 has the entryname tape_06.acs. Each ACS has the suffix acs.

access isolation mechanism (AIM)
The mechanism used to guarantee that only authorized persons access certain classes of information.

accounting start up (acct_start_up.ec)
An exec_com that creates and initializes most of the accounting directories and data bases. The accounting start-up exec_com is run in the admin mode when a new system is created.

accounting update
The process of computing resource usage for each logged in user, saving it for later use by accounting routines, and logging out any user whose usage exceeds the limit specified by the project administrator. This function is performed by the answering service at a site-specified interval (every 15 minutes, by default).

alias
An alternative to a Person_id or Project_id that a user can type when logging in. It may be up to eight characters long, must begin with a lowercase letter, and must be unique at the site. The typical value for a Person_id alias is the user's initials. Not every Person_id and Project_id need an alias. Aliases are intended as a convenience for Person_ids and Project_ids with long, hard-to-type names. Since giving every Person_id an alias would double the size of the PNT hash table, the system administrator should establish some policy for the assignment of aliases.

anonymous user
A user who is not registered as an identifiable person on the Multics system. Projects that allow anonymous users have an asterisk (*) value for a personid keyword statement in the PMF.

answering service
A subsystem that runs in the initializer process (Initializer.SysDaemon.z); responds to requests from users to be logged in and logged out; responds to requests from administrators for system table installations; executes commands typed by the operator; and performs accounting updates.

background process

An absentee process created in response to a request in one of the background absentee queues (queues 1 through 4).

billing

The process of computing the total resource usage of each user and each project, preparing bills and other reports addressed to various administrative personnel, and then resetting the usage figures to zero. Billing is run by the accounting administrator. It is recommended that billing be run at the end of each month, on or before the last day of the month. Terminology in documentation and program output (e.g., "month-to-date charges") assumes this recommendation is followed. If a billing period longer than 90 days is attempted, several programs will malfunction.

canonicalization routine

The conversion of a terminal input line into a standard (canonical) form. This is done so that lines that appear the same on the printed page, but that may have been typed differently (i.e., characters overstruck in a different order), appear the same to the system.

communications link

A link leased from a common carrier which provides the medium for transmission of data through a data communication network.

crank

The absentee job that performs various accounting functions. The usage figures gathered by the accounting updates are copied, by the crank, into data bases that are used in billing. Month-to-date summaries are produced. Projects past their cutoff limits are cut off (users on these projects are no longer permitted to log in). The printing of cutoff warning messages is based on the usage figures computed daily by the crank. It is recommended that the crank be run daily near the end of operations. Terminology in documentation and program output (e.g., "daily report") assumes this recommendation is followed. Commands executed by crank are kept in the master.ec.

daemon

One of several system service processes that perform such tasks as process creation, backup, network control, and printing segments on the line printer. Daemons are usually registered on either the Daemon or SysDaemon projects and have the daemon attribute in the PDT so they can be logged in by the operator via the message coordinator.

device class

A mechanism for separating output according to access class is the "device class". Each request type can be partitioned into any number of separate device classes. Each device class can have a range of access classes specified for it. One or more devices can be specified for each device class.

disk quota

The maximum number of pages that can be used in a hierarchy of directories. Each user is allotted a predetermined amount of quota; however, quota can be increased by a system administrator.

driver

A system-supplied program module that can manipulate a device that you can define in the I/O daemon tables. An *input* driver receives files and subtrees from the output driver or from a local or remote device. An *output* driver transmits files and subtrees to the input driver or to a local or remote device. See "Understanding I/O Daemons" later in this manual.

foreground process

A process created in response to either a user's login request or a request in the foreground absentee queue.

initializer process

The first process (Initializer.SysDaemon.z) created during system initialization. This process remains in existence until the system is shut down and runs several subsystems critical to system operation.

interactive process

A process created in response to a user's login request.

I/O interfacer (IOI)

The facility of the supervisor through which user programs (via I/O modules) can operate peripheral devices. IOI provides for the operation of the I/O hardware and the multiplexing of channels and other physical resources between processes. IOI can only be used to manipulate a device once a process has acquired the right to use that device via Resource Control Package (RCP).

limited service subsystem (LSS)

A subset of the Multics system imposed on users by the project administrator.

line description

A line is the name of a logical communication line_id in the I/O daemon table and denotes the beginning of a line description. "Channel", "att_desc" and "device" are three substatements associated with a line description. These substatements are contained in the I/O daemon table.

load control group

A specific number of projects that share guaranteed access to the system. (See "Managing System Load/Allocating Processor Resources" later in this Section.)

local device

A device that is connected to the input/output manager (IOM) hardware with cables. See "Setting Up I/O Daemons" later in this manual.

logical volume

A set of physical volumes that are always mounted together. (See "Managing Storage System Disk Volumes" later in this Section.)

logical volume registration record

When a logical volume is created a registration record is created for it. This registration record contains:

- a list of the physical disk volumes comprising the logical volume
- the logical volume identifier (name and/or uid)
- public or private switch
- the list of directories that would be legitimate to put on this particular logical volume

major device

A combination device that contains more than one logical device (e.g., a printer and a punch in a single physical unit). A major device is connected to Multics via a single communications channel; it can be attached by one process only.

message coordinator

A subsystem that runs in the initializer process and routes messages between one or more daemon processes and one or more terminals being used as operator consoles.

message segment

A special type of segment that is managed by Multics supervisor programs and is not directly accessible to the user. A message segment is simply a permanent place to hold interprocess messages, e.g., dprint and dpunch requests.

minor device

Multiple subdevices such as a printer and a punch at a remote job entry (RJE) station.

multiplexer channel

There are two types of multiplexer channels: an X.25 multiplexed channel and a HASP multiplexed channel. An X.25 channel is bidirectional and operates with subchannels that can both send and receive information. A HASP channel is unidirectional and has subchannels that either send or receive data.

password

A character string supplied by a user and known only to him and the software that controls access to the system. It may be from one through eight ASCII printing characters including backspace, but excluding space and semicolon. It is supplied with the user's Person_id at login time to validate the identity of the user.

Person_id

A name assigned to each user of the system, which must be unique at the site. It is usually some form of the user's name (usually his surname). The name must be from one to 20 alphanumeric characters long and usually begins with a capital letter. A user has only one Person_id regardless of the number of projects on which he is registered.

physical volume

A disk pack. Sometimes the combination of pack and disk drive is referred to as the physical volume. A physical volume is divided into records of 1024 words each. (See "Managing Storage System Disk Volumes" later in this manual.)

private logical volume
A logical volume whose access is restricted; its physical volumes are usually not permanently mounted. See "Managing Storage System Disk Volumes" later in this manual.

process
A program or group of programs in execution: an address space and an execution point. Each logged-in user has his own process. (See the *Multics Programmer's Reference Manual*, Order No. AG91.)

process overseer
A procedure called during process initialization that sets up the environment. It then calls the listener to start reading commands.

programmer number
An optional number (e.g., an employee number) that can be associated with a Person_id. The number may be from one to sixteen digits long and is used for external record keeping purposes.

project
A set of users grouped together for accounting and access control purposes.

Project_id
The name assigned to a project. The name must be from one to nine alphanumeric characters long, must begin with a capital letter or a digit, and must be unique at the site.

public logical volume
A logical volume whose access is unrestricted; its physical volumes are usually permanently mounted. (See "Managing Storage System Disk Volumes" later in this manual.)

quota
The limit on the storage space occupied by the files in each directory. Quota is measured in records, with one record being equal to one page. Each user is allotted a predetermined amount of quota; however, the system administrator can increase a user's quota. (See "Managing Quota" later in this manual.)

RCP resource management (RCPRM)
An optional part of the Multics operating system that manages the use of peripheral I/O devices (such as tape drives, printers, and punches) and physical volumes that can be mounted on these devices (such as tape reels, forms, and disk packs). These resources are managed by programs located in the administrative ring (ring 1) and run in the user's process. The RCPRM facility handles registration and acquisition of resources, which includes deregistration and release.

registry
The main data base of the Resource Management option of RCP (called RCPRM). Registration of a resource provides information such as the type of resource, the name of the resource and in what access class range you can use the resource. (See "Managing I/O Resources - RCPRM" later in this manual.)

remote device
A unit record device which is connected to the FNP and communicates with the IOM via the FNP (see *Operator's Guide to Multics*, Order No. GB61).

remote driver

A program that embodies specific knowledge of how to manipulate a particular device. A remote driver should be specified for all remote printer/punch/reader stations. A remote driver uses logical devices to control the physical devices. (See *Operator's Guide to Multics*, Order No. GB61.)

request type

Part of the I/O daemon table definition of a device. The request_type statements are found in the I/O daemon table. See "Understanding I/O Daemons" later in this manual.

resource

A component of the system, such as a tape drive, a tape volume, CPU time, or secondary storage, whose use is controlled by RCP and can be charged for.

resource control package (RCP)

The software, operating in ring 1, that controls the reservation, assignment, and use of the resources whose resource types are described in the RTDT.

resource price

An entry in the price lists given in the installation_parms segment, giving the price of some system resource, such as CPU time, special forms, tape drive usage, etc. Note that the resources controlled by RCP are a subset of the resources whose prices are given in the installation_parms segment.

ring

A level of privilege at which programs may execute. Lower numbered rings are more privileged than higher numbered rings. The supervisor program runs in ring 0; most user programs run in ring 4. (See *Multics Programmer's Reference Manual*, Order No. AG91.)

root logical volume (RLV)

The special logical volume that contains all of the files and directories needed to bring Multics up to ring 4 command level. Ring 4 is the second ring the initializer process enters when the system is started up. The RLV is the only logical volume that contains directory segments.

root physical volume (RPV)

The single physical volume required to bring Multics up to ring 1 command level, and perform ring functions like reloading volumes and performing some kind of crash recovery. The RPV also contains the root directory, ">", as one of its segments.

shift

System usage may be divided into a maximum of eight shifts, numbered from 0 to 7. These shifts may begin at any half-hour during a week; shift numbers always remain the same unless they are changed in the shift table.

time-record product (TRP)

The amount of time that a record is in storage. The time-record product is the basis for charging for disk storage. The time-record product is also referred to as the time-page product or tpp.

user

A person or logical entity, such as a daemon, who is registered on the Multics system and, therefore, has the ability to log in. Each user is associated with a project and is identified for access control purposes by the concatenation of his Person_id, Project_id, and tag. A person may be registered as a user on more than one project; thus one person can be two or more different users (since a user is identified by the combination of his Person_id, or Project_id, and tag).

User_id

An access control name of the form Person_id.Project_id.tag. Since the tag portion is rarely explicitly given, User_id is often defined as a Person_id.Project_id pair.

user subsystem

A collection of programs that provide a special environment for some particular purpose, such as editing, calculation, or data management. It may perform its own command processing, file handling, and accounting. A subsystem is said to be closed if:

1. All necessary operations can be handled within the subsystem

2. You are unable to use the normal Multics environment from within the subsystem

work class

A group of processes that are guaranteed (by the priority scheduler) a specified percentage of the available virtual CPU time. A work class can be an entire load control group, a sample user or a class of users such as all I/O daemon or all absentee jobs in a given queue. There may be up to 16 work classes named 1 through 16. Their definitions are stored in the MGT and modified by the ed_mgt command. (See "Managing System Load/Allocating Processor Resources" later in this manual.)

PART III
CONTROLLING RESOURCES

SECTION 6

UNDERSTANDING RESOURCES

Multics includes a series of resources that individuals can use (I/O devices) or consume (computer time and storage). Multics lets you allocate system resources and charge users for their use accordingly. The following system resources can be manipulated to control usage and/or charges for their use:

- Interactive CPU time
- Interactive real time
- Absentee CPU time
- Absentee memory units
- I/O daemon usage
- Disk storage
- Channels

The Resource Control Package (RCP) lets you control access and use to the following I/O resources:

- Devices
- Volumes

RCP software operates in ring 1 and controls the reservation, assignment, and use of devices and volumes that you define in the resource type description table (RTDT).

INTERACTIVE CPU TIME (MACHINE TIME)

Interactive CPU time is the amount of time a user's programs and interactive applications require for processing by the Central Processing Unit. Interactive CPU time can be controlled in the following ways:

- System load control parameters specified in >sc1>installation_parms.
- Values specified in the Master Group Table (MGT).
- Use of accounting procedures to set the price on interactive CPU time according to the shifts that you define for your site.

INTERACTIVE REAL TIME (CONNECT TIME)

Interactive real time (or connect time) is a measure of the user's login time; that is, the time a user actually spends on the system from login to logout. Interactive real time is managed by:

- Values in the installation_parms segment and the Master Group Table
- A user's login attributes (primary or secondary, bump or nobump).

ABSENTEE CPU TIME

Absentee CPU time is a measure of the amount of central processing unit time used by a user's absentee (noninteractive) processes. The absentee processes can be either foreground absentee processes or background absentee processes. Foreground absentee and background absentee CPU time can be managed with:

- Values specified in installation_parms and the MGT

- The SAT keywords:
  - Max_background
  - Max_foreground
  - Abs_foreground_cpu

ABSENTEE MEMORY UNITS

Absentee memory units are a measure of the usage a user makes of system memory resources while executing absentee processes (background or foreground). The amount of memory an absentee process can use is also a function of the the SAT keywords described above.

I/O DAEMON USAGE

I/O daemon usage is a measure of the users's use of the system process that is responsible for printing and punching. Daemon usage is managed by values specified in installation_parms as defined in the iod_tables.iodt.

DISK STORAGE

The amount of disk storage allocated to a user and the volumes the user is allowed to access can be managed by:

- Quota allocation
- Device access control segments located in >sc1

For more information on quota management see Section 13.

CHANNELS

A user's access to a specific channel is controlled by channel access control segments in >sc1>rcp and values specified in the Channel Master File. For a complete description of communication channel security, see Section 20.

DEVICES AND VOLUMES

Device and volume management is under the control of the Resource Control Program (RCP) and Resource Management (RCPRM). Refer to Section 7 and Section 8 for a complete description of the RCP facility.

SECTION 7

MANAGING I/O RESOURCES--RCP

The Resource Control Package (RCP) provides a mechanism for device reservation, assignment, and attachment.

The functions reserve, assign, and attach are organized into hierarchical levels. Defaults are provided at each level so that users not desiring to exercise features specific to a level do not have to concern themselves with that level.

```
1  reserve
   2  assign
      3  attach
      3  detach
   2  unassign
1  cancel
```

The first level involves the reservation of resources by processes. Tape drives, disk drives, tape volumes and disk volumes can be reserved. Reservations are process-specific and remain in effect until the process requests a cancellation. Reservation implies that a process temporarily has exclusive rights to a resource. This exclusive right means that no other process can use that resource for the duration of the reservation. Reservation does not necessarily imply that a resource is actually being used. Multiple resources can be reserved with one reservation.

Assignment, like reservation, is process-specific and lasts until unassignment or process termination. Only devices can be assigned. An assignment also gives a process temporary exclusive rights to a device. Assignment does not necessarily mean that a device is currently being used. That is the function of the next level, attachment. Only one device can be assigned per assignment.

A resource cannot be used until it is attached. When RCP is called to attach a resource, it initiates communication with the ring-0 subsystem that actually provides the use of the resource. Before the attachment is completed, RCP performs all initialization necessary to allow the attaching process to begin using the resource. For devices, this involves attaching the device via IOI and making sure that the device is ready and that any volume needed is accessible, mounted, and authenticated.

The hierarchical relationship among reservation, assignment, and attachment implies that a higher-level function (e.g., reservation) can stand alone. Often, higher-level functions completely perform the lower-level function. For example, an attachment of a device implicitly assigns the device if it is not already assigned. Assignment, however, does not cause a reservation to occur. RCP can perform the following device reservation, assignment, and attachment functions:

1. Reserving a resource. This means that no other process can use it until the reservation is cancelled.

2. Explicitly assigning a reserved device. The device is assigned to a process but is not attached.

3. Attaching an explicitly assigned device.

4. Attaching an unassigned device. Since a device cannot be attached until it is assigned, RCP automatically assigns the device and then performs the attachment. The device is said to be implicitly assigned.

5. Detaching an implicitly assigned device. After the device is detached, RCP automatically unassigns the device.

6. Detaching an explicitly assigned device. The device is detached but is not unassigned.

7. Explicitly unassigning a device. If the device is attached, it is first detached and then unassigned.

8. Canceling reservation of a resource.

The rules stated above imply that I/O modules do not have to be concerned with the assignment or unassignment of devices. They need to be concerned only with the attachment and detachment of a device. RCP, however, does allow the above rules to be overridden. When detaching a device an I/O module can tell RCP to retain the device assignment regardless of whether the device was explicitly or implicitly assigned.

When a process terminates, RCP automatically detaches and unassigns all devices currently assigned to that process and cancels any reservations for that process.

The reservation of resources and cancellation of reservations is accomplished from command level via the reserve_resource and cancel_resource commands or by using the -resource control argument with the enter_abs_request command. The explicit assignment and unassignment of devices is done from command level via the assign_resource and unassign_resource commands. The listing of reservations, assignments, and attachments is done from command level via the list_resources command. The commands are described in the *Commands and Active Functions* manual, Order No. AG92.

CONTROLLING ACCESS TO DEVICES

An important feature of RCP is its ability to control access to the various devices that it manages. It does this through the use of access control segments (ACS). RCP searches the >scl>rcp directory for ACS segments.

An access control segment is a zero length segment that is associated with the object to be protected. The access control list (ACL) and ring brackets of the access control segment determine the user's access to the RCP device or communications channel. The access control segments for RCP devices must have the form <device_name.acs>.

ACCESS TO SYSTEM DEVICES

System devices, generally belong to the system for the benefit of all users. For users to access devices, the ACL for the device's access control segment should be set as follows:

```
sa device_name.acs rw User_id.Project_id.tag
```

In addition to setting the appropriate access to the ACS, use the set_ring_brackets command (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64) to set the appropriate ring brackets for the device.

SETTING UP RCP

RCP is always active on the system and the only action required on your part to use RCP is the creation of the access control segments to control access to various devices.

UNDERSTANDING THE RESOURCE TYPE MASTER FILE

The Resource Type Master File (RTMF) is a ASCII file that describes all of the resource types on your system. You must convert the RTMF to a system-usable binary version called the Resource Type Definition Table (RTDT) by using the cv_rtmf command and then install it using the install command (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64).

The RTDT, which is generated by the RCP administrator, defines all of the resource types known by the system. Also defined in the RTDT are default values for the potential attributes, the potential access class range, and the charge type to be used in billing for resources of a given type. The registries contain information specified by the administrator at registration time or when a resource is acquired for a user. The RTDT and the registries are described in subsequent sections.

SECTION 8

MANAGING I/O RESOURCES - RCPRM

The Resource Management feature of RCP permits the implementation of AIM control on devices and volumes. Resource Management permits the use of access control segments with volumes as well as devices.

Resource Management is an optional facility which offers the ability to retain registration information for all resources that it controls. It does this by providing administrative interfaces for the registration of resources. Registration of a resource provides information such as: what type of resource this is, what its name is, which attributes it possesses, or in what access class range the resource can be used. Once a resource is registered, users may acquire it; system administrators can also acquire it to a user (or to the system pool) at the time it is registered. The act of acquisition makes a user the owner of the resource (i.e., liable for all charges to that resource and in control of discretionary access to the resource).

The hierarchical level of RCP and RCPRM functions is illustrated below.

```
1  register
                                              Resource Management
   2  acquire
**************************************************************************
      3  reserve

         4  assign

            5 attach
                                                    Resource Control
            5 detach

         4  unassign

      3  cancel
**************************************************************************
   2  release
                                                    Resource Management
1  deregister
```

A variety of information is stored by the system as part of Resource Management. This information is under the control of the RCP administrator. This includes all of the information in the resource type description table (RTDT) and most information in the registries.

SETTING UP RCPRM AND RCPRM MODES

To enable Resource Management (RCPRM), an RCP mode is provided that you can enable with the ed_installation_parms command (see the *Administration, Maintenance and Operations Commands* manual) using the rsc_mgmt_enabled keywords. It is important to note that the use of resource management is not automatic; it must be turned on.

The automatic registration of resources can also be enabled using ed_installation_parms but it is not recommended at sites running AIM.

Once resource management is enabled, resources can be acquired by users. If auto registration is not on, you must register resources before they can be acquired by users.

Prior to initiating Resource Management with ed_installation_parms, the following procedures should be followed:

1. Prepare an exec_com that will register all devices.

2. Prepare a complete list of all existing tape volumes and their owners. The final list should be in the form of an exec_com that can be used to call register_resource to register and acquire tape volumes to their respective owners.

3. Turn on rsc_mgmt_enabled (rsc_) in installation_parms. authentication (auth) should be set to nominal. auto_registration should remain off.

4. Reinstall the RTDT. This will create the registries (see Section 10).

5. Register all devices (an ec can be used)

6. Register all tapes (an ec can be used)

7. Shutdown and reboot the system. From now on, the system can only mount registered tapes.

REGISTRATION AND ACQUISITION

Registering a resource with the register_resource command (see the *Administration, Maintenance and Operations Commands* manual) sets the owner to "free". If the potential access class range is not specified, it is set by default to the value in the Resource Type Description Table (RTDT) for that resource type. If potential access class range is specified, it must be a subset of the access class range in the RTDT for that resource type.

Once the owner field in the registry is set to something other than "free", the resource is considered to be acquired. An acquired resource also has an actual access class range (see below). Acquisition can be done by the RCP Administrator when you register the resource by specifying the -owner argument with the register_resource command (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64). It is recommended that all devices be acquired at registration time with "owner=system".

Users acquire free resources with the acquire_resource command. Acquiring a resource establishes an accounting owner for the resource. This can only be done if you are an RCP Administrator (i.e., e access to rcp_admin_ gate).

The access class range is set during acquisition to the current authorization provided that it is within the potential access class range already in the registry.

USING ACCESS CONTROL LISTS WITH RCPRM

RCPRM uses access control segments to enable the use of ACLs on devices and volumes.

An access control segment is a zero length segment that is associated with the object to be protected. The ACL and the ring brackets of the ACS segments determine the user's raw mode to the RCP resource.

The access control segments for RCP resources must have the form <resource_name>.acs. When the resource management facility of RCP is *not* enabled, an access control segment can exist for devices only (not for volumes). These access control segments must be created and stored in the directory >sc1>rcp. When the Resource Management facility of RCP *is* enabled, the access control segments for RCP resources (devices and volumes) must be created and can reside anywhere. The owner of the resource specifies the ACS pathname in the registry during the acquire operation, or by using the set_resource command. It is the responsibility of the resource owner to create the ACS segment.

USING THE AIM MECHANISM WITH RCPRM

The Resource Management option of RCP permits the administrator to assign AIM access class ranges to the device/volumes.

Access class ranges are used by RM to specify that a user within a range of authorizations can use a particular resource.

An access class range is simply a pair of AIM access classes separated by a colon. The first value of the pair is the minimum access class and the second is the maximum access class. If only a single access class is specified when an access class range is expected, the minimum and maximum access class values are both the same (i.e., a range of one value). The second access class of the pair (the maximum) must be greater than or equal to the first (the minimum).

Access class ranges are specified at the time the device/volume is registered and must be a subset of the one specified in the RTDT for resources of that type. The actual access class range is determined at acquisition time. See the register_resource command in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

CLEARING RESOURCES

Resources on which the manual_clear attribute has been set to "yes" in the RTDT must be "cleared" (degaussed) of all residual information before they can be used by another user. To clear the tape for other users, use the clear_resource command.

RELEASING LOCKS ON RESOURCES

If the -lock control argument is on when a tape volume is registered with the register_resource command, users are not allowed to release the resource. This value is off by default and, usually, users are allowed to release tapes after use to be returned to the free pool or "lending library" of tapes.

If a lock is set on a tape volume and you want to release the lock, use the set_resource command with the -lock control argument set to "off". For detailed information, see the set_resource command.

MANAGING AN RCPRM FREE TAPE POOL

The pool of tapes kept for use by the general user community is usually called the free tape pool. As the manager of the free pool you can:

- Add tapes to the pool
- List the tapes in the pool
- Remove tapes from the pool

The procedures for performing each if these functions is described below.

Adding Tapes to the Pool

A pool of tapes is kept available to be acquired by users via the acquire_resource command. To add tapes to the pool, use the register_resource command, as follows:

```
register_resource tape_vol TAPES -potential_attributes
   len=LEN
```

For example, to add the tapes UT0001-UT0005, type:

```
rgr tape_vol UT0001 UT0002 UT0003 UT0004 UT0005 -pattr
   len=600
```

Listing Tapes in the Pool

To list the tapes in the free pool, use the list_resources command. For more information on the list_resources command (see the *Commands and Active Functions* manual, Order No. AG92).

Removing Tapes from the Pool

To remove tapes from the pool, use the deregister_resource command, as follows:

```
deregister_resource tape_vol TAPES
```

Only tapes that are currently free can be removed from the pool.

SETTING/RESETTING USAGE LOCK AND LOCATION FIELDS

When a tape is removed from the machine room temporarily, you can turned on its "usage lock". This prevents the user from trying to mount the tape and from releasing the tape. In addition, the location field may be set to indicate that the tape is not in the machine room, for example:

```
setr tape_vol &2 -lock on -location "Not in machine room"
    -priv
```

When the tape is returned, the usage lock and location fields should be reset:

```
setr tape_vol &2 -lock off -location "" -priv
```

GETTING INFORMATION ABOUT A PARTICULAR TAPE OR GROUP OF TAPES

To get detailed information about a particular tape, either a free tape or a tape that has been acquired by a user, use the resource_status command as follows:

```
resource_status tape_vol TAPENAME -priv
```

To get detailed information about a group of tapes, resource_status can be used in conjunction with the list_resources active function as follows:

```
resource_status tape_vol [list_resources -acq -type tape_vol
   -user *.*] -priv
```

SECTION 9

MANAGING THE RESOURCE TYPE MASTER FILE

GENERAL FORMAT OF THE RESOURCE TYPE MASTER FILE

The Resource Type Master File (RTMF) consists of a series of entries, one for each resource type that is known to the system. Each entry consists of a series of statements and substatements of the form:

```
<keyword>:  <parameter>;
```

PL/I-style comments beginning with /* and ending with */ may appear anywhere within the RTMF. Similarly, blanks, tabs, and newlines not embedded within a keyword or parameter are ignored. However, to include blanks, tabs, newlines, colons, or semicolons in a parameter, it must be enclosed in quotes. If a parameter begins with a quote, all immediately following characters up to the next quote are taken as the parameter. It is not possible to embed quotes within a quoted string in the RTMF.

The last statement in the RTMF must be the statement:

```
end;
```

Entry Format

Each entry in the RTMF describes one resource type known to your system. An entry includes the name of the resource type being described (i.e., tape_drive, disk_vol, printer); fixed parameters for that resource type (density, track, etc), and default values. Resource types can be listed with the list_resource_types command.

The default values can be omitted during the registration process. For example, if you want the potential access class to be "system_low:system_high" for tape_drives, then setting this in the RTMF permits you to omit this option during registration of the tape_drives. The potential access class in the registry for the tape_drive uses the default found in the RTMF. Furthermore, if a potential_access_class is specified during registration of the resource, its value must be within the range specified in the RTMF (see "Application of Defaults" below).

Resource Type Entries

The RTMF consists of one or more complete resource type entries. Each entry must include the name of the resource type being described, fixed parameters for that resource type, and default parameters to be applied at registration time in the absence of explicit parameters. In addition, a resource type entry may contain one or more sets of special registration parameters.

Each entry must begin with one of the following statements:

```
Volume:  <volume_type_id>;
Device:  <device_type_id>;
```

All statements following, until the next occurrence of a Volume or Device keyword, apply to this entry.

Fixed Resource Parameters

The following statements describe fixed parameters that apply to all resources of the given resource type, and may not be changed at registration time:

Limit: <number>;
Limit: open;
defines the maximum number of this type of resource that any user is allowed to assign at one time. If open is specified, no limit is enforced. This limit applies only to resources for which the system is the accounting owner. The limit does not apply to users using privileged RCP facilities. If this statement is not supplied, open is assumed.

Attribute_domain: {<attribute_list>};
specifies all of the allowable attributes that any resource of this type may possess. The <attribute_list> is of the form:

```
<name1>,<name2>,...,<nameN>
```

The attribute list is allowed to be empty. In entries for a volume type, any attribute may be followed by an asterisk. This specifies that any volume for which this attribute is currently active can be mounted only on a device which also possesses this attribute. See "Naming Rules for Attributes" below for more about the consequences of naming attributes.

Canonicalizer: <virtual_entry>;
specifies a program which is to be used to perform standardization of resource names as typed by the user, before they are presented to RCP Resource Management. If this statement is omitted, no canonicalization is performed. See "Canonicalization Routines", below.

Manual_clear: <yes_or_no>;
controls the operation of the resource data security features of automatic acquisition and release. If <yes_or_no> is the string "yes", volumes of this type are locked (when released by an accounting owner) in a way that does not allow another user to acquire them until the operator certifies that the volume has been cleared of all residual information. If this statement is omitted, "no" is assumed.

Default Resource Parameters

The following statements describe default parameters that apply during registration of all resources if no values for the parameters are explicitly provided in the registration command:

potential_attributes: <attribute_list>;
specifies the default potential attributes with which a resource is registered if no potential attributes are explicitly provided. The syntax of the attribute list is as given above for the Attributes keyword, except that asterisks on attributes are unnecessary.

access_range: <aim_range>;
specifies the default potential access class range with which a resource is registered if no potential access class is explicitly provided. If <aim_range> contains delimiters such as commas, colons, or spaces, it must be enclosed in quotes.

attributes: <attribute_list>;
specifies the default initial attributes with which a resource is registered if no attributes are explicitly provided. These attributes are also used to provide any defaults for attributes specifications at the time a resource is used (i.e., track=9 if no track specification is present).

Canonicalization Routines

Each resource type entry in the RTMF may specify an associated canonicalization routine. This routine is responsible for canonicalizing resource names as typed by the user into standard forms (for example, stripped of leading zeroes). Standard canonicalization routines exist and are supplied for certain resources (see the table below). Sites may choose to replace or augment the standard algorithms with those of their own choosing (for example, to enforce a more restrictive set of rules for tape volume names).

The canonicalization routine for a resource type is specified as an entry in the "Canonicalizer" statement in an RTMF. (See the documentation for cv_entry_ in the *Multics Subroutines and I/O Modules* manual, Order No. AG93) for a more detailed description of the syntax of virtual entries.) The canonicalization routine specified should be installed in a system library and must be executable from ring 1, as it will be used by RCP Management in that ring.

The calling sequence for all canonicalization routines supplied by the site must conform to the following:

```
declare <virtual_entry> entry (char(*), char(*), pointer, fixed bin(35));

call <virtual_entry> (input_name, output_name, info_ptr, code);
```

input_name
is the resource name to be canonicalized. (Input)

output_name
is the canonicalized representation of input_name. (Output)

info_ptr
is currently unused and is null. (Input)

code
is a standard error code. (Output)

Standard Canonicalization Routines

Resource Type	Routine Name
tape_vol	canon_resource_name_$tape_vol
tape_drive	None
punch	None
reader	None
console	None
printer	None
disk_vol	None
disk_drive	None
Special	None

Special Registration Parameters

You can specify, by name, a special class of any resource, and establish defaults for this class that are different from the normal defaults. A statement of the form:

```
type:  <type_name>;
```

introduces a special class of the resource named <type_name>. All default resource parameters following, until the next occurrence of a type, Volume, or Device keyword apply to this special class.

Resource Type Synonyms

Each additional name must be defined via a Volume or Device keyword, as above, followed by the statement:

```
Like:  <volume_type_id>;
```

or

```
Like:  <device_type_id>;
```

where <volume_type_id> or <device_type_id> specifies the resource type for which this entry is to be a synonym. For example, if the resource type "tape" is to be made synonymous with resource type "tape_vol", the RTMF should contain an entry of the form:

```
Volume:  tape;
Like:    tape_vol;
```

No other keywords should appear in a synonym definition entry.

Naming Rules for Attributes

Attributes provide a description of a volume or device that assists the resource management facility in the proper matching of volumes with compatible devices. To produce correct combinations, attribute names must comply with the set of rules described below.

Attributes may be grouped or ungrouped. Grouped attributes specify a set of properties applicable to a device or volume such that only one attribute of that set can be currently active at any given time. For example, a reel of tape may have potential attributes that allow it to be recorded at densities of 556, 800, or 1600; however, at any given time, the data on it is in only one of those densities. Grouped attributes have names of the form:

```
<identifier>=<value>
```

For example, the attributes mentioned above are named "den=556", "den=800", and "den=1600". This notation allows RCP to recognize that any request to make one of these attributes the current attribute of a device or volume also implies that all other attributes in that grouping must be made inactive.

When adding or changing an attribute in a string of attributes, all attributes in the string must be respecified or else existing attributes are nullified by the change. Also, any attribute string must contain a value for each grouped attribute. For example, if the attribute domain includes "track=..., model=..., and den=...," the device you are setting the attributes for (or registering) must contain values for each grouped attribute.

Ungrouped attributes have simple names, such as "trainok" (to specify that this device accepts a removable print train) or "building_12" (to specify that this device or volume is located in building 12).

Application of Defaults

When you register a resource, that resource can be registered using the defaults for the registration parameters that are specified in the RTMF. Alternately, you can explicitly specify parameters for which defaults can also be specified in the table, such as attributes and AIM classes. If any such parameter is explicitly specified, the corresponding default for that parameter is overridden. It should be noted that any parameter specified must be a subset of the default value specified in the RTMF. For example, if you specify -pacc during the registration of tape_drive tapf_01 as "system_low:system_high" and the default values for access_range are "system_low:system_low" in the RTMF, an error message is produced.

When you register the resource, any default parameters defined for that resource type are applied in the absence of a corresponding explicitly specified parameter.

If you register the resource with the "-type <subtype_name>" control argument, any default parameter defined for the special class named <subtype_name> is applied in the absence of a corresponding explicitly specified parameter. In the case of duplicate resource type and special class parameters, the special class default parameters override the general resource type parameters. In addition, any default parameters specified for that resource other than those defaults in the special class are applied.

If no special classes of a resource are defined, and the defaults for the resource are not all present, it is always necessary for the missing parameters to be explicitly specified for every registration request for a resource of this type.

If special classes of a resource are defined, then defaults within the definition of special classes can be used either to replace corresponding defaults specified for the resource in general, or to supplement for missing defaults that are not specified for the resource in general.

In the latter case, you cannot perform a simple default registration of the resource, but must either specify the missing items explicitly in the command line, or use the "-type <subtype_name>" control argument to take advantage of the additional defaults provided in a special class.

Understanding Reserved Resource Names

RCP uses the information in the RTDT to decide what types of resources are known to the system, how they are to be handled, and what important attributes they possess. In the initial implementation, sites can use this flexibility to augment the standard complement of attributes for certain resources.

For example, a site with tape drives in more than one location can register these drives with an additional simple attribute, thereby allowing users to request assignment of a tape drive in the remote location. Additionally, the tape reels in the remote location can be tagged with a matching attribute, marked in the RTDT as requiring that attribute of its tape drive.

Although this mechanism is very flexible, the necessity of having certain standard and reserved resource type names and attribute names cannot be avoided. Standard software (e.g., tape and disk I/O modules) needs to refer to a domain of resources by standard names, as well as certain attributes of the resources. Since these strings must be the same at all sites, certain resource types and certain resource attributes must be contained in all RTMFs. The cv_rtmf command checks for their existence and refuses to process an RTMF that lacks them. This list of required resource type names and attributes is also found in the include file, rcp_mandatories.incl.pl1.

RCP does not allow the name "scratch" to be used in registering a resource. A scratch tape is one of the unmarked tapes in an unreserved pool that is used for "scratch" (i.e., no information is saved on it from session to session. After every use, it is demounted and returned to the system pool).

Reserved Resource Names

The following resources are mandatory and must appear in all RTMFs:

```
Device:   disk_drive
Device:   tape_drive
Volume:   tape_vol
Volume:   disk_vol
```

Understanding Reserved Attribute Names

The following attributes are mandatory for the devices named, and must appear in all RTMFs:

For the disk_drive device:

```
model=400       model=451       model=181
model=191       model=500       model=402
```

For the tape_drive device:

```
track=7         track=9
den=200         den=556
den=800         den=1600
model=400       model=500
model=600       model=610
```

Adding a Resource Type

If you want to add a resource type to the RTMF, you must edit the RTMF with one of the Multics text editors and enter the information manually. The procedures for adding a resource to the RTMF are:

1. Invoke a text editor.

2. Edit the RTMF adding the appropriate information for the resource according to the RTMF syntax specified above.

3. Convert the RTMF to the RTDT with the cv_rtmf command as described above.

4. Install the RTDT with the install command.

Changing a Resource Type

To change a resource type, follow the same procedures as described for adding a resource. You must invoke a text editor and change the RTMF manually, convert the RTMF to the RTDT, and install the RTDT.

Deleting a Resource Type

To delete a resource type, follow the same procedures as described above for adding a resource type or changing a resource type.

UNDERSTANDING THE RESOURCE TYPE DEFINITION TABLE

The Resource Type Definition Table (RTDT), as mentioned, is the binary version of the RTMF. The RTMF must be converted to the RTDT via the cv_rtmf command for the table to be usable by the system. Once you have converted the RTMF into the RTDT, install the RTDT using the install command.

Converting the RTMF to the RTDT

Each time you modify the RTMF, you must convert it to the RTDT with the cv_rtmf command. To convert the RTMF to the RTDT, type:

```
cv_rtmf rtmf_path {-control_args}
```

For details on the cv_rtmf command, type "help cv_rtmf" or see the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Installing the RTDT

Once you have converted the RTMF to the RTDT with the cv_rtmf command, you must install the RTDT with the install command. To do this, type:

```
install rtdt_path {-control_args}
```

For details on the install command type "help install" or see the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Printing the Contents of the RTDT

To display the contents of the RTDT use the display_rtdt command as follows:

```
display_rtdt {type1 ... typen} {-control_args}
```

For details on the display_rtdt command, type "help display_rtdt" or see the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Sample Resource Type Master File

This RTMF includes the following volume types: tape_vol and disk_vol; and the following device types: tape_drive, disk_drive, reader, punch, printer, console, and mpc.

```
Volume:              tape_vol;
Attribute_domain:    len=2400,len=1200,len=600;
Implies:             tape_drive;
Manual_clear:        no;
Canonicalizer:       >sss>canon_tape_name_;

potential_attributes:
                     len=2400,len=1200,len=600;
attributes:          len=2400;
charge_type:         tape_vol;
access_range:        "system_low : system_low";

/* --------------- */

Device:              tape_drive;
Attribute_domain:    track=7,track=9,model=400,model=500,
                     model=601,model=610,den=200,den=556,
                     den=800,den=1600,den=6250,speed=75,
                     speed=125,speed=200;
Accepts:             tape_vol;
Manual_clear:        no;
Canonicalizer:       ;          /* No canonicalization. */

potential_attributes:
                     track=9,model=610,den=800,den=1600,
                     den=6250,speed=200;
attributes:          track=9,model=610,den=1600,den=6250,
                     speed=200;
charge_type:         tape_drive;
access_range:        "system_low : system_low";

/* --------------- */

Device:              punch;
Attribute_domain:    model=201,model=300,model=301,model=401;
Accepts:             ;
Canonicalizer:       ;

potential_attributes:
                     ;
attributes:          ;
charge_type:         punch;
access_range:        "system_low : system_low";
```

```
/* --------------- */

Device:              reader;
Attribute_domain:    model=201,model=301,model=401,model=500;
Accepts:             ;
Canonicalizer:       ;

potential_attributes:
                     ;
attributes:          ;
charge_type:         reader;
access_range:        "system_low : system_low";

/* --------------- */

Device:              console;
Attribute_domain:    model=EMC,model=IBM,model=LCC;
Accepts:             ;
Canonicalizer:       ;

potential_attributes:
                     ;
attributes:          ;
charge_type:         console;
access_range:        "system_low : system_low";

/* --------------- */

Device:              printer;
Attribute_domain:    model=301,model=1000,model=1200,
                     model=1600,speed=1000,speed=1150,
                     speed=1200,speed=1600;
Accepts:             ;
Canonicalizer:       ;

potential_attributes:
                     ;
attributes:          ;
charge_type:         printer;
access_range:        "system_low : system_low";
```

```
/* --------------- */

Volume:              disk_vol;
Attribute_domain:    model=181*,model=191*,model=400*,
                     model=402*,model=451*,model=500*,
                     model=501*,use=io*,use=ss*;
Implies:             disk_drive;
Manual_clear:        no;
Canonicalizer:       ;

potential_attributes:
                     model=500,use=ss,use=io;
attributes:          model=500,use=io;
charge_type:         disk_vol;
access_range:        "system_low : system_low";

/* --------------- */

Device:              disk_drive;
Attribute_domain:    model=181,model=191,model=400,
                     model=402,model=451,model=500,
                     model=501,use=io,use=ss;
Accepts:             disk_vol;
Canonicalizer:       ;

potential_attributes:
                     model=500,use=ss,use=io;
attributes:          model=500,use=io;
charge_type:         disk_drive;
access_range:        "system_low : system_low";

/* --------------- */

Device:              special;
Attribute_domain:    ;
Accepts:             ;
Canonicalizer:       ;
potential_attributes:
                     ;
attributes: ;
charge_type:         special;
access_range:        "system_low : system_low";

/* --------------- */

end;
```

SECTION 10

MANAGING I/O RESOURCE REGISTRIES

UNDERSTANDING REGISTRIES

A registry is a multisegment file. There is one registry for each resource type identified in the RTDT. Each registry contains one entry for each registered resource of that type.

To make a resource of a particular type available to the system you have to register the resource. Information about the resource is kept in the registry for that resource type. A resource can be registered with information such as its name, owner, access class range for AIM restrictions, and other information. Registering a resource makes the resource part of the "free pool" of resources available on the system. A resource that is registered in the free pool is available for acquisition by everyone. Resources that are registered remain registered until they are explicitly deregistered which makes them unavailable to the system.

REGISTERING A RESOURCE

To enter a resource in the registry, use the register_resource command (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64). Once registered, a resource's information can be manipulated via the set_resource, acquire_resource, release_resource, and deregister_resource commands. Use of the -priv control argument requires "e" access to either rcp_admin_, rcp_priv_, or rcp_sys_ depending upon the command used.

When you register a device with the register_resource command, the registration process informs the system that the resource is available for users who are authorized to access it. Use of the -owner option results in an implicit acquire. It is recommended that devices be registered and acquired with owner=system. The registration process also defines the resource with specialized control arguments listed below:

- attributes
- type
- name
- unique id
- potential attributes
- potential access class
- release lock
- owner
- access class
- attributes
- allocation flag
- ACS pathname

Each of these categories is discussed below. For detailed information on the register_resource command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Attributes

The -attributes control argument defines the current values of the resources attributes. It can only be used when -owner is also specified (i.e., implicitly acquire). It is limited to those values specified in the Potential Attributes for this resource type. If the Potential Attributes have not been explicitly set, they are by default the attributes string defined in the RTDT.

Attributes provide a description of a volume or device that assists the resource management facility in the proper matching of volumes with compatible devices. To produce correct combinations, attribute names must comply with the set of rules described below.

Attributes may be grouped or ungrouped. Grouped attributes specify a set of properties applicable to a device or volume such that only one attribute of that set can be currently active at any given time. For example, a reel of tape may have potential attributes that allow it to be recorded at densities of 556, 800, or 1600; however, at any given time, the data on it is in only one of those densities. Grouped attributes have names of the form:

```
<identifier>=<value>
```

For example, the attributes mentioned above are named "den=556", "den=800", and "den=1600". This notation allows RCP to recognize that any request to make one of these attributes the current attribute of a device or volume also implies that all other attributes in that grouping must be made inactive.

When adding or changing an attribute in a string of attributes, all attributes in the string must be respecified or else existing attributes are nullified by the change. Also, any attribute string must contain a value for each grouped attribute. For example, if the attribute domain includes "track=..., model=..., and den=...," the device you are setting the attributes for (or registering) must contain values for each grouped attribute.

Ungrouped attributes have simple names, such as "trainok" (to specify that this device accepts a removable print train) or "building_12" (to specify that this device or volume is located in building 12).

When defining attributes for a volume type, any attribute may be immediately followed by an asterisk. If this notation is used as a current attribute in the description of any particular volume, the device chosen for this volume at mount time must possess this attribute as a potential attribute. For example, the attribute "track=7*" occurring in the definition of volume type "tape" specifies that any tape reel with the current attribute of "track=7" may only be mounted on a tape drive possessing the potential attribute "track=7."

The following attributes are mandatory for the devices named, and must appear in all RTMFs:

```
For the disk_drive device:

model=400          model=451          model=181
model=191          model=500          model=402

For the tape_drive device:

track=7            track=9
den=200            den=556
den=800            den=1600
model=400          model=500
model=600          model=610
```

Charge Type

The type must be defined as the resource type in the RTDT when registering this device.

Name

The name of the resource must be a unique name that identifies the resource.

Potential Attributes

Potential attributes defines the values of attributes that may be set for this tape. These values are limited to those defined in the RTDT. By default, it is set to the potential_attributes defined in the RTDT. A site may define other, possibly more meaningful, attributes. Potential attributes may be changed, added, or deleted at any time. However, old attributes remain in the RTDT (marked as deleted) and there is a limit of 72 for the number of attributes that may be in the RTDT.

Potential Access Class

The potential access class sets the AIM access class parameters. Users at any authorization within the access class range are allowed to red and write to the resource.

Release Lock

The release lock is used to prevent a tape from being released by a user. It is off by default. This lock may be considered as the definition of whether or not a tape is part of the "lending library". Any tape with this lock off is potentially available to anyone, if its owner releases it.

Owner

The owner is the person who owns the resource, usually a tape. It may either be an actual user in the form Person.Project, "free", or "system". If the owner is "free" the resource does not belong to anyone. It is part of the "lending library" and is available to be acquired (and thus owned) by any user using the acquire_resource command. Access to this resource is rw for everybody.

If the owner is Person.Project, the tape belongs to that user and only to that user. The default access for the tape is rew for the owner (Person.Project.*) and null for everyone else. If the tape is released (via release_tape), its owner field becomes "free".

If the owner is "system", the tape is owned by the system for the common benefit of all to use. These tapes are referred to as the "system pool". The default access for these resources is rw for everybody. RCP translates the volume name "scratch" into a request for an appropriate volume from the system pool. Some system programs use the name "scratch", such as T&D's. Therefore, at least one tape must be registered with "system" as the owner. This feature can be useful for tapes to be used for a site's plotter or PPS. One can do this by registering tapes with special attributes, e.g., "pps", and having programs request tapes named "scratch" with "-attributes pps".

Access Class

The access class is the initial AIM access class range. Users at any authorization within the access class range are allowed to read and write to the resource (provided they meet other access requirements).

Allocation Flag

Allocation state is a bit switch. It is off by default.

Access Control Segment Pathname

The access control segment pathname is the pathname of the access control segment (ACS) for the resource. By default there is no ACS for the resource. To change the ACS for a resource, an ACS pathname must be specified, the ACS segment created, and its ACL set appropriately.

THE REGISTER_RESOURCE COMMAND

Use the register_resource command to make a particular resource known to the system. The registration process informs the system that there resource is available for users who are authorized to access it. For details on the register_resource command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

For example, to add tapes UT0001 to UT0005 to the tape pool, type:

```
register_resource tape_vol UT0001 UT0002 UT0003 UT0004 UT0005
   -pattr len=600
```

To acquire tapes when they are registered, use the register_resource command with the -owner control argument, as follows:

```
register_resource tape_vol <tape_names> -potential_attributes
   len=<tape_length> -owner Person.Project -release_lock on
```

For example, to register 2400 foot tapes VD0001 to VD0004 to be used by the volume dumper:

```
rgr tape_vol vd0001 vd0002 vd0003 vd0004 -rll on -pattr
   len=2400 -ow Volume_Dumper.Daemon
```

AUTOMATIC REGISTRATION

To allow for a more expedient transition from RCP without Resource Management to RCP with Resource Management, an RCP mode is provided that may be turned on or off with the ed_installation_parms command, using the auto_registration and rsc_mgmt_enabled keywords. It is important to note that the use of resource management is not automatic--it must be turned on.

DEREGISTERING A RESOURCE

To remove tapes from the pool, use the deregister_resource command.

```
deregister_resource tape_vol TAPES
```

Only tapes that are currently free can be removed from the pool.

To release and deregister tapes that have been assigned, use the release_resource command with the -priv argument and then the deregister_resource command, as follows:

```
rlr tape_vol TAPENAME -priv;drr tape_vol TAPENAME
```

MAKING CHECKPOINT COPIES OF REGISTRIES

Part of Resource Management is registry management. Every time that a change is made in a registry, an entry describing that change is made in a journal. The Resource Management journal contains a record of all operations performed on each registry since the time represented by the last checkpoint. The checkpoint copy is a complete copy of all the registries. The system accounting facility automatically makes a checkpoint copy of the system registries every night using the copy_registry command. When the checkpoint copy is made, the journal entries are deleted since they are no longer needed. (The -reset control argument to the copy_registry command empties the registry journal of entries.)

RECOVERING DAMAGED REGISTRIES

If, for any reason, the registries become damaged, there are several procedures you can follow to ensure their recovery. Before attempting a full scale recovery of the registries, determine if the registries are definitely damaged, do the following:

1. Turn off the damage switches for the tape registry by typing:

```
lsdrb >scl>rcp>tape_vol.rcpr 4 4
st >scl>rcp>tape_vol.rcpr>*
lsdrb >scl>rcp>tape_vol.rcpr 1 1
```

2. List the contents of the registry by typing list_resources and resource_status commands. If this works, the registry is not damaged and recovery procedures do not have to be undertaken. If this fails, perform the steps listed below.

3. Get the damaged registry out of the way by using remove_registry to rename it from tape_vol.rcpr to tape_vol.old.

```
remove_registry >sc1>rcp>tape_vol
```

4. Copy the checkpoint copy of the registry into place. Do this with the copy_registry command.

5. Shutdown, reboot. The reconstruct_registry command must be run by the Initializer before the answering service is started.

```
boot stan
admin
```

6. Reconstruct the registry using reconstruct_registry by typing:

```
reconstruct_registry >sc1>rcp>tape_vol
```

7. Delete the old registry by typing:

```
delete_registry >sc1>rcp>tape_vol
```

8. Leave admin mode and finish booting the system.

DELETING A REGISTRY

The remove_registry command can be used to remove a registry from service. This command should only be used in exceptional circumstances. The activity of removing a registry is normally reserved to the Initializer process which automatically removes a registry when a new RTDT is installed that no longer contains a resource type associated with that registry. In general, manual removal of registries is only necessary in the process of recovering from a catastrophic system failure and reload, where the existing registries and RTDT are out of agreement.

This command requires access to the rcp_sys_ gate.

Once the registry has been removed by the remove registry command, it can be deleted with the delete_registry command (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64).

SECTION 11

SETTING PRICES FOR RESOURCES

FIGURING OUT WHAT RESOURCES SHOULD COST

The calculation of the proper price for the use of a resource has been the subject of considerable refinement. Each site has its own way of computing these rates. One possible method is to list all costs involved in running the service. This includes programmer and operator salaries, cards and paper, hardware rental, and modems. Desired profit or planned loss could be included. These costs are then attributed to cost pools, one for each resource that is chargeable. Resources that cannot be attributed to the cost of providing a particular service, such as operator salaries or the motor generator, are included in the category "overhead," and the cost of this pool is spread proportionally over the others. Each pool's cost is then divided by estimated customer usage, to obtain a "break-even price." These figures are then manipulated to make reasonable prices. If an estimate is uncertain, it is probably wise to round up strongly. For shift differentials, the following table has been used:

```
shift 1   115% of break-even
shift 2   90%
shift 3   50%
shift 4   80%
```

The shift prices should be rounded up to the nearest multiple of $.50 per minute, just to make the prices easy to remember.

The default CPU price for all shifts is $240.00/hour; for connect time, $1.25/hour; and for memory units, $15.00/1000 units.

SETTING RATES IN INSTALLATION_PARMS

Most site parameters used in the operation of the administrative system are kept in >sc1>installation_parms. When the accounting start up exec_com (acct_start_up.ec) is executed during start up at a new Multics site, it automatically sets these parameters to default values. The system administrator uses the ed_installation_parms command both to read current values and to alter these values in the installation_parms segment. A discussion of each parameter follows.

It is recommended that when reading this information for the first time, invoke the ed_installation_parms command to print out the >sc1>installation_parms segment to be used for reference.

Interactive CPU Time

The rate structure set for interactive CPU time is kept in the segment called "default" in installation_parms. You can change it using the ed_installation_parms command (see the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64). The default for interactive CPU time for all shifts is set a $240.00/hour.

Interactive Real Time

The rate structure set for interactive real time is kept in the segment called "default" in installation_parms. You can change it using the ed_installation_parms command (see the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64). The default charge for interactive real time for all shifts is set at $1.25/hour.

Interactive Memory Units

The rate structure set for interactive memory units is kept in the segment called "default" in installation_parms. You can change it using the ed_installation_parms command (see the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64). The default charge for interactive memory units for all shifts is set at $15.00/1000 units.

I/O Daemon Usage, Absentee CPU Time, and Absentee Memory Units

The daemon rates and absentee rates are also kept in the installation_parms segment; the rates may be inspected and modified by the system administrator by specifying the queue_prices parameter with the ed_installation_parms command. There are a maximum of four background absentee queues, and four daemon queues for each request type. Absentee CPU time prices are stored by queue, as are default I/O daemon rates per 1000 lines. For example:

```
Queue     Abs CPU    Abs mem    I/O Daemon

1          110.00     00.00      1.00
2          110.00     00.00      0.75
3          110.00     00.00      0.50
4          110.00     00.00      0.50
```

The default values, which apply to all queues, are:

```
220.00     15.00      1.80
```

Disk Storage

The disk storage price can be set with the "retype prices" keyword of ed_installation parms.

Registration Fee

The registration fee for the person_id in the project can be set with the "process" keyword of ed_installation_parms.

Device Prices

The device_prices keyword of ed_installation_parms can be used to manipulate the prices (dollars per hour for each shift or per use for tape mounts and disk mounts) for each of up to 16 miscellaneous devices (teletype channels, high-speed channels, tapes, etc.). For the print request, if the name of a device is typed ahead on the input line after the device_prices keyword, the prices for that device are printed; otherwise the prices of all devices are printed. For the retype request, the name of each device is typed and new prices are requested for all shifts.

Resource Prices

The "resources" and resource_prices keywords of ed_installation_parms can be used to set prices for resources.

SETTING RATES IN THE IOD_TABLES

The line_charge substatement in the i/o daemon tables defines the resource price names for the line charge of each queue of the request type. This substatement is optional. If not specified, default values are used. If specified, each price name must be defined in the system price table, or the compilation of the the iod_tables will fail. The price names for each queue must be given in order, from queue 1 to the maximum number of queues for the Request_type description. Each price is defined in units of dollars per 1000 lines, or dollars per page.

UNDERSTANDING RATE STRUCTURES

You can define up to 10 distinct sets of prices, called rate structures, at your site. The rate structures are numbered 0-9, and each site specifies names for them (up to 32 characters). All commands that set or display rate structure information identify the rate structures by their names; the accounting database identifies the rate structures by their numbers.

Programs that display actual charges (resource_usage and all of the billing programs) also display the name of the rate structure that was used to compute those charges. The dprint tailsheet also shows the rate structure used to calculate charges. This information is printed on the line below the total cost of the dprint request. At sites with no rate structures defined other than the default one, the printing of the rate structure name is suppressed. A rate structure other than the default may be specified either when creating a project using the new_proj command, or by changing attributes of an existing project with the edit_proj command. You can change the prices in existing rate structures or add new rate structures using the ed_installation_parms command.

The procedure for setting up a new rate structure is:

1. Define it, using the ed_installation_parms command,

2. Wait for the next answering service startup (at which time its existence will be made known to all the programs that do charging),

3. Install an SAT that has the new rate structure specified for some projects.

An attempt to install an SAT that uses the new rate structure before the next startup will be rejected.

ASSIGNING PROJECTS TO RATE STRUCTURES

You assign each project to any site-defined rate structure, and the project is charged according to the prices established in that rate structure. A rate structure is set for a project as whole; separate rate structures for the individuals on a project are not allowed. All programs that estimate and display an individual user's charges (e.g., general_ready) automatically use the rate structure in effect for that user.

By default, only one set of prices is defined (rate structure 0) and each project is assigned to rate structure 0. By default, the name of the default rate structure in installation_parms is "default". Sites have the option of changing this name.

CHARGING FOR VOLUME DUMPER SERVICES AND VOLUME RETRIEVALS

Volume dumping is charged as part of the basic cost for storing disk pages (disk charges).

There is no automatic method for charging for volume retrievals. It is suggested that volume retrievals be charged as miscellaneous charges on a manual basis.

PART IV
MANAGING THE STORAGE SYSTEM HIERARCHY

SECTION 12

UNDERSTANDING SYSTEM DIRECTORIES

A directory is a catalog of the segments "beneath" it. The directory contains information about all the attributes of each segment. Every segment and directory in the system (except the *root directory*, the originating directory of the Multics system) is immediately under another directory.

Any segment or directory can be located by its entry in the directory immediately superior to it. The *directory* concept in the Multics System is the key to several Multics features including storage structure, administrative control, search rules, and naming conventions. A single directory hierarchy is used for both system and user segments. Figure 12-1 shows the basic structure (at the upper level of the storage hierarchy) assumed by the Multics system. Additional segments and directories can be created at this level of the structure as well as at lower levels. This section contains a description of the most significant system directories.

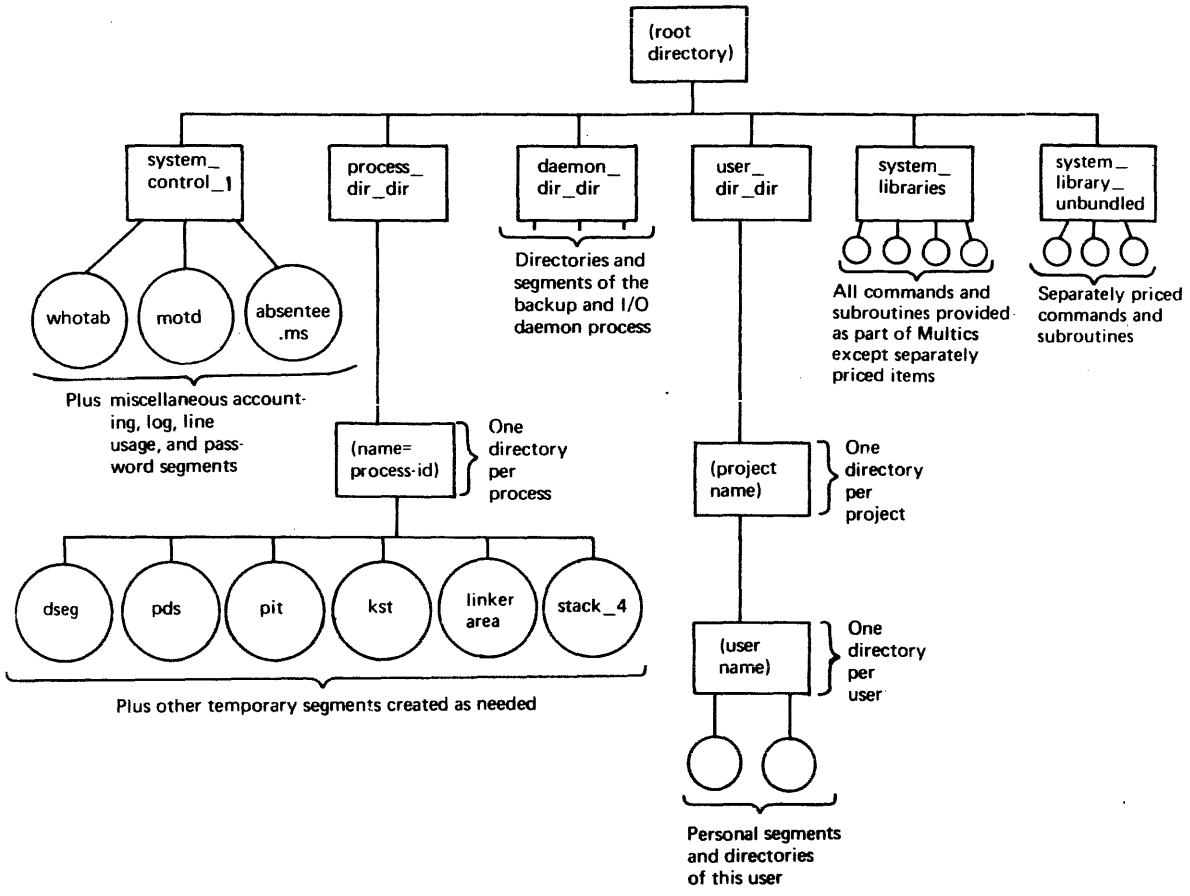

Figure 12-1. Multics Directory Hierarchy

CONTENTS OF DAEMON DIRECTORY DIRECTORY (>ddd)

Daemon processes are organized in a hierarchy; the main node of this hierarchy is the >deamon_dir_dir (>ddd). The >ddd directory contains segments and directories used to support the various system daemon processes. A daemon is a system service process that performs such tasks as storage system backup, process creation and network control. The daemon_dir_dir contains the following directories:

- >cards
  a storage pool for card deck image segments read by the system card input process.

- >gcos
  used by the GCOS daemon

- >io_daemon_dir
  holds all I/O daemon data bases

These directories and their contents are described in the following paragraphs. The access isolation mechanism (AIM) class for all these directories is system low.

Contents of the >cards Directory

The storage pool for card deck image segments is headed by >ddd>cards. Each access class has its own directory contained in the cards directory. Storage is always allocated within the access class directory that corresponds to the caller's authorization. A person directory contains all segments and multisegment files for a given person at a given access class. Person directories are contained in the appropriate access class directory. A person directory is created for each person who needs temporary storage. For example, if a user with person_id TSmith is at system_low, the following directory is allocated for his card deck image segments:

```
>daemon_dir_dir>cards>system_low>TSmith
```

Contents of the >gcos Directory

This directory is used by the GCOS daemon. The GCOS daemon is a facility designed to aid in the simulation of a GCOS environment on Multics. The daemon allows standard GCOS jobs to be submitted from IMCV magnetic tapes in GCOS standard system format or from a card reader. For more information on the >udd>GCOS directory, refer to the *Multics GCOS Environment Simultator* manual, Order No. AN05.

Contents of >io_daemon_dir Directory

The >daemon_dir_dir>io_daemon_dir directory contains a set of administrative data bases and working storage used to direct the activities of the I/O coordinator process and the various device drivers.

The main data base, iod_tables, is set up by a system administrator. Most of the other segments and directories are created and maintained by the I/O coordinator acting on information contained in the iod_tables segment.

The following segments are contained in io_daemon_dir: (>idd)

coord_comm.ms
: ring 1 message segment in which driver processes place messages for the I/O coordinator.

coord_lock
: a segment used by the process overseer of IO.SysDaemon to prevent initialization of a driver process before an I/O coordinator has been created; also prevents creation of more than one I/O coordinator.

iod_tables
: master control tables for the I/O coordinator; output segment produced by the iod_tables_compiler command on request of a system administrator.

iod_tables.iodt
: source segment for iod_tables; may be updated by a system administrator and compiled to yield iod_tables.

iod_working_tables
: working copy of iod_tables used by the device drivers and users; copied from iod_tables during I/O coordinator initialization.

iodc_data
: a segment containing the process identifier of the I/O coordinator and the event channel identifier to be used by a new driver for initial communication with the coordinator.

printer_notice
: an optional segment containing information that the site administrator wants to be printed on the page following the head sheet of every printer listing, local or remote. The segment must contain ASCII text. It should be no longer than 60 lines and the line length should correspond to the shortest printer device in use. This feature is useful in notifying users of printing charge rates, available request types, when they are processed, stock forms used for each, and other useful information covering the printing operations on the system.

XXX_N.ms
: ring 1 message segment for I/O daemon queues; one such message segment is created for each priority queue of a request type; XXX is the request type name and N is the queue number (1<= N <=4).

For information on the recommended access for the above segments, refer to Section 18.

The ddd>idd also contains the site dependent directories described below; a directory is created and named for each active device.

CONTENTS OF >udd>idd>cord

This directory is working storage for the I/O coordinator, which is created and managed by the I/O coordinator.

CONTENTS OF >ddd>idd>io_msg_dir

This directory contains mailboxes for each device (station) for which driver to driver messages will be sent or received. For more information refer to the *Multics System Maintenance Procedures* manual, Order No. AM81.

CONTENTS OF >ddd>idd><majordevice>

There is a separate directory for each major device currently being run by a device driver. The I/O coordinator manages these directories; the names of these directories are site dependent. Each major device directory contains a driver status segment for each minor device associated with the major device. The access class is the authorization of the device driver.

CONTENTS OF >ddd>idd>rqt_info_segs

An administrator creates this directory to hold any request type info segments used by drivers. For more information on this directory, refer to the *Multics System Maintenance Procedures* manual, Order No. AM81.

CONTENTS OF >ddd>volume_backup

The >ddd>volume_backup is the repository for the volume backup system's databases. The volume backup system scans the physical volumes used by the storage system.

CONTENTS OF THE DOCUMENTATION DIRECTORY (>>doc>info)

This directory contains segments, such as the info segments, documenting the use of Multics. This directory is not shown on Figure 12-1.

CONTENTS OF THE DUMPS DIRECTORY (>dumps)

The >dumps directory contains system dumps copied by the copy_dump command. Additionally, the dumps directory contains system, FNP dumps, and images of stacks and directories released by the directory salvager. It also contains the directory >dumps>safe_pdirs where dead processes are put by the copy_deadproc command.

CONTENTS OF LIBRARY DIRECTORY DIRECTORY (>ldd)

The >ldd directory contains files used to generate the Multics system. It is the directory of the subtree in which the source and object archives for system library programs are stored.

CONTENTS OF LOGICAL VOLUME DIRECTORY (>lv)

The >lv directory contains supervisor data bases used for registration and access control of logical volumes and master directories. The >lv directory is not shown on Figure 12-1. Two supervisor subsystems, *logical volume management* and *master directory control*, use these data bases to manage the logical volume environment. These subsystems are implemented in ring 1.

Logical volume management maintains a list of all logical volumes registered on the system, and their attributes. Logical volume management knows, for example, which physical volumes belong to a logical volume. When a request to mount a logical volume is received, the subsystem cooperates with ring 0 volume management to ensure that all required physical volumes are mounted. (The supervisor runs in ring 0.)

Master directory control manages directories whose segments reside on logical volumes other than the root logical volume (RLV). Such directories are called master directories. A master directory is simply the point where the hierarchy "branches out" to another logical volume.

The supervisor data bases reside under the directory >lv either as segments or links to segments elsewhere in the hierarchy, as described below.

There is one segment for each logical volume known to the system. The attributes that are maintained include the physical volumes that constitute the logical volume. Each segment (or link) has the names:

lv.LVNAME	where LVNAME is the name of the logical volume
lvid.LVID	where LVID is a character representation of the unique identifier of the logical volume
pv.PVNAME	where PVNAME is the name of a physical volume belonging to the logical volume (there is one for each such physical volume)
pvid.PVID	where PVID is a character representation of the unique identifier of a physical volume belonging to the logical volume (there is one for each such physical volume)

The volume registration segments are used by ring 1 volume management in the initializer process. They are ring 1 segments with rw access to the initializer and general r access.

CONTENTS OF THE PROCESS DIRECTORY DIRECTORY (>pdd)

This directory contains one directory for every process currently in the system. The name of an individual process directory is derived from the unique identification of the process. This directory stores temporary segments created by a process. These segments are only retained for the life of the process that creates or uses these segments.

When a process is created, a process directory is established with the six initial segments listed below:

```
process data segment (pds)
descriptor segment (dseg)
known segment table (kst)
stack_n
process initialization table (pit)
<unique_name>.area.linker
```

process data segment (pds)
: A supervisor data base, the pds keeps a record accessible only to the supervisor.

known segment table (kst)
: A supervisor data base, the kst contains the correspondence between segment numbers and segments known to the associated process. This segment is accessible only to the supervisor.

process initialization table (pit)
: The pit contains information that is used to initialize the process. This information includes the user_id, home directory, attributes, and accounting data.

descriptor segment (dseg)
: A supervisor data base, the dseg contains the correspondence between segment numbers and their absolute memory addresses and access permissions. This segment is accessible only to the supervisor. It is always segment 0.

stack_n
: This segment contains the stack used for PL1 automatic variables, subroutine call, and return operations. One stack segment is created for each active ring. The last character of the stack name is the ring number. The ring 0 stack is kept in system_library_1. Many of the other stacks, such as the ring 4 stack (user stack) are kept in the process directory.

<unique_name>.area.linker
: This segment, managed by linker, contains interprocedure links and PL/I internal static storage. If the total requirements for linkage information and static storage exceed the length of a segment, additional segments are created as needed under a similar name. Each active ring has its own linkage segment. The linkage area also contains other system storage (e.g., external static and FORTRAN common).

Other segments that are created by various Multics subsystems and editors are also commonly found in the process directory.

CONTENTS OF THE RELOAD DIRECTORY (>reload_dir)

This directory contains reload maps produced by the reload command. The Volume_Maps is an array that is kept on each physical volume. The >reload_dir is not shown on Figure 12-1. The array contains one entry for each physical 1024-word record of the volume.

CONTENTS OF THE SITE DIRECTORY (>site)

A special directory called >site has been created for use by the customer site.

Sites purchasing Emacs may place a start_up.emacs segment in this directory for users who do not have their own Emacs start up segment. Sites purchasing Forum should follow the installation instructions given in the *Multics Forum Interactive Meeting System User's Guide* Order No. CY74.

Sites using Data Management should see Section 43, "Data Management Administration", for information on setting up the Data Management hierarchy under >site.

CONTENTS OF THE SYSTEM CONTROL DIRECTORY (>sc1)

The most important directory for the system operations and critical programs is the Initializer's home directory:

>system_control_1 >(sc1)

This directory contains segments referenced by the Initializer (system control) process while it is:

1. Logging in a user
2. Logging out a user
3. Accounting for user usage
4. Starting up the system

The contents of the >sc1 directory include the following special segments:

absentee_user_table
: a table that has an entry for each absentee process, with the name, accounting data, process identification, and arguments for each absentee job that is running.

absentee_foreground.ms
: the foreground absentee queue (a ring 1 message segment).

absentee_N.ms
: background absentee queue N (a ring 1 message segment).

admin.ec
: an exec_com segment containing sequences of commands that the operator can execute by using the exec command. For additional information on the contents of admin.ec refer to the *Multics Administration, Maintenance, and Operations Commands* manual. Order No. GB64.

answer_table
: a table that has an entry for each interactive process, with the name, accounting data, and process identification of the logged-in user.

cdt
: the channel definition table (CDT), containing an entry for each terminal channel, specifying per-channel attributes.

daemon_user_table
: a table that has an entry for each daemon process, with name, accounting data, process identification, and source identifier for each daemon entry currently logged in.

DEVICE_NAME.queue
: a queue used to distinguish between a segment named "c.h006.queue" and a segment named "device_name queue" (see "Volume Registration Segments" in this section).

installation_parms
: a table containing site settable parameters, including shift definitions and prices.

>sc1>as_logs>log
: a log of events of interest to the initializer, many initializer messages typed to the operator are also recorded in this segment.

login_help
: a segment that is typed if a user types "help" instead of "login".

mc_anstbl
: the message coordinator's table of terminal channels. To view this table, use the command mc_list.

message_of_the_day
: a segment containing messages addressed to all users; edited by the operator command, message, or any text editor; printed either by the default start_up.ec at log-in time or by the commands, "help motd" or "print_motd."

rate_structure_N
: where N is a digit in the range 0 through 9. The segment rate_structure_N contains the pricing information for the Nth rate structure defined in installation_parms. By default, a site uses only a single rate structure, number 0. This rate structure is contained in installation_parms, and therefore, the name rate_structure_0 is an additional name on the segment installation_parms. If the site defines new rate structures, additional rate_structure_N segments are created to contain the pricing information.

mgt
: the master group table (MGT) specifying the guaranteed load units for each load control group and listing the current occupancy of all groups. It is accessed at each login to determine whether the attempted login would overload the system; it also contains the minimum percentage of system resources to be given to each work class and determines the work class membership of each load control group

MRT
: the message coordinator's message-routing table. This table is automatically maintained by the system.

>sc1>syserr_log>syserr_log
: permanent log of system errors and events of interest.

PNT
: the person name table (PNT), contains the Person_ids of all registered persons except anonymous users. It is referenced at all logins to check the user's password and to check for a default Project_id, person_authorization, and password. It is also referenced at card input time by the card input process, including RJE process.

  Note that, while portions of the user entry in the PNT are stored in encrypted form, any encryption algorithm is susceptible to a sophisticated, computer-assisted code-breaking effort. Therefore the System Administrator should ensure that access to the PNT is as restricted as possible. In general, only the SysAdmin and SysDaemon projects should have access to the PNT.

rtdt
: an installed table (a binary segment) that defines the resource types used by RCP.

sat
: the system administrator table (SAT), containing an entry for each registered project plus some per-system quantities; referenced at every login to validate the user's Project_id.

sat.ht
: a hash table for the SAT; this table helps to speed up the search in the SAT.

shift_config_change.ec
: an exec_com segment that is invoked at each shift or configuration change.

stat_seg
: system statistics, sampled at every accounting update (normally every 15 minutes) by the as_meter_ program and copied daily by the copy_as_meters command (in the crank).

syserr_log
: contains the syserr log segments.

system_start_up.ec
: an exec_com segment that is invoked automatically when the answering service is started.

ttt
: the terminal-type definition table. Contains the definitions of all valid terminal types recognized by the system.

vcons_tab
: the message coordinator's virtual console table. This table is automatically maintained by the system.

whotab
: the public list of logged-in users.

XXX.message
: the message coordinator input queue for source XXX.

XXX.queue
: the message coordinator output queue for the device whose channel name is XXX.

For information on the recommended access for the above segments, refer to the "Assuring the Security of the File System" later in this manual.

In addition, other segments may be kept in the >sc1 directory, such as segments necessary for system reloads. These segments include the following:

- Initializer.SysDaemon
- Card_Input.Daemon
- SysDaemon
- SysAdmin
- Operator
- SysMaint
- Daemon

Other important directories in >sc1 include the following:

- admin_acs
- as_logs
- mc_acs
- pdt
- rcp
- syserr_log
- update
- volume_backup_accounts

The *admin_acs* directory contains the following access control segments:

- bump_user.acs
- process_termination_monitor.acs
- tandd.acs
- sat.install.acs
- mgt.install.acs
- rtdt.install.acs
- cdt.install.acs
- ttt.install.acs
- absentee_proxy.acs

The as_logs directory contains various logs including the answering service log and "lg", which contains information about all user and daemon logins and logouts.

The *pdt* directory contains a project definition table (PDT) for each registered project and accounting data for each user of the project. Whenever a user logs in, his entry is located in his project's PDT, and the user's attributes are used to initialize his process.

The *rcp* directory contains access control segments for peripheral devices controlled by RCP.

The *syserr_log* directory contains syserr messages or hardcore messages, produced by the Multics supervisor and some of the online programs.

The *update* directory is used when a project administrator requests the installation of a new copy of the PDT.

CONTENTS OF SYSTEM LIBRARIES

The system libraries are described below.

>system_library_standard (>sss)
: This directory contains the libraries of Multics commands and subroutines. The commands are documented in the *Multics Commands and Active Functions* manual, Order No. AG92. The subroutines are documented in the *Multics Subroutines and I/O Modules* manual, Order No. AG93.

>system_library_1 (>sl1)
: This library contains a small set of subroutines that are reloaded each time the system is reinitialized. This directory also contains all hardcore commands and subroutines provided as part of Multics. Separately priced items are not included.

>system_library_unbundled (>unb)
: This directory contains all the separately priced (unbundled) software products supported on Multics such as COBOL or Emacs. The inclusion of this directory is optional.

>system_library_auth_maint (am)
: This directory contains private commands and subroutines provided by programmers at the local installation site. The inclusion of this directory is optional.

>system_library_tools (>tools)
This library contains software primarily of interest to system programmers.

CONTENTS OF THE USER DIRECTORY DIRECTORY (>udd)

This directory contains all the project directories and directories of users registered on these projects.

It is the base of a subtree containing all of the personal segments of individual users. The immediate contents of user_dir_dir is a set of directories, one for each project that uses Multics. These are called project directories and they usually contain one personal directory for each user working on that project.

The home directories of users are often called user directories.

CONTENTS OF SYSADMIN DIRECTORY

Another important directory is the system administration's project directory:

```
>udd>SysAdmin
```

The home directories for each system administrator registered on the SysAdmin project are kept here. In addition, there are two other important directories contained in >udd>SysAdmin: the admin directory contains all of the local site accounting data bases; the lib directory contains the local system-administration library.

CONTENTS OF >udd >SysAdmin>admin Directory

All data segments pertaining to system administration are kept in the directory:

```
>udd>SysAdmin>admin
```

This directory is the working directory for the accounting administrators (who have no access to the regular programming tools of the system and no private segments except a mailbox). This directory contains the following:

PNT.safe.pnt
: a copy of the installed PNT (which may include password changes made by users online), saved every day; used in case the system copy is destroyed.

billing_footnote
: segment of announcements (e.g., forthcoming price changes) to project supervisors that is printed at the bottom of each project's usage summary report (on the mailing copy only); this segment is optional.

bwchart.print
: the system "black and white" chart. showing system availability for the week; printable report.

CMF.cmf
: the channel master file, which is compiled by the cv_cmf command, producing CMF.cdt.

CMF.cdt
: binary channel definition table; installed in >sc1 as "cdt" using the install command.

cutrpt
: a report of accounts that are nearing the cutoff date or have actually been cut off. This report is prepared daily.

daily_log_N
: printable report describing general system status, followed by entries from the system log, selected by the crank using the *.ssl segments and prepared for printing each day (N is a number from 0 to 9. e.g., daily_log_2).

daily_report.control
: a segment that describes which projects are summarized under each category in the daily and monthly statistical reports; used by the reset_use_totals command during monthly billing to set up categories for the next month.

disk_stat
: a binary segment giving disk quota and usage for all directories with quota; prepared daily.

diskreport
: a report of project disk usage, prepared and dprinted daily by the dodrp.absin job.

installation_parms
: a backup copy of the segment in >sc1 that has the same name, saved weekly; used in case the system copy is damaged. This table contains site-settable parameters, including shift definitions and prices.

*.ssl
: segments used by the daily_log_process program to select messages from the log to be placed in the daily_log_N segments for printing.

MGT.mgt
: a version that is edited before installation; this segment is edited by the ed_mgt command and installed in >sc1 as "mgt" using the install command.

meter_data
: a copy of the contents of >sc1>stat_seg for use in generating system statistical reports; produced daily by the copy_as_meters command.

miscfile
: the miscellaneous charges and credits journal.

monthly_usage_and_revenue.report
: a segment containing the monthly usage and revenue report; produced when monthly billing is run.

old.admin.dir_info
: a summary of the directory >udd>SysAdmin>admin, saved weekly and compared with the previous week's summary.

old.pdt.dir_info
: a summary of the directory >sc1>pdt, saved weekly and compared with the previous week's summary.

old.sc1.dir_info
: a summary of the directory >sc1, saved weekly and compared with the previous week's summary.

pmf.archive
: an archive of all PMFs for undelegated projects (for undelegated projects, the current PMF; for delegated projects, the PMF at the time the project was delegated).

projfile
: the project registration segment, containing descriptive information about each project, including title, supervisor name and address, and disk storage usage.

reqfile
: the requisition segment, containing account number, billing address, total charges, month to date charges since project was created, and requisition number for each usage account.

RTMF
: an ASCII segment compiled by the cv_rtmf command to produce RTMF.rtdt.

RTMF.rtdt
: binary resource type description table; installed in >sc1 as "rtdt" using the install command.

safe_projfile
: a copy of the projfile segment in >udd>SysAdmin>admin, saved weekly; used in case the projfile is damaged.

safe_reqfile
: a copy of the reqfile segment in >udd>SysAdmin>admin, saved weekly; used in case the reqfile is damaged.

smf.cur.sat
: a copy of the system SAT; this segment is edited by the edit_proj and new_proj commands and installed in >sc1 as "sat" using the install command.

sumry
: a printable report of project accounting status, prepared daily.

system.report
: the daily statistical report: first part of daily_log_N.

system_start_up.ec
: a copy of the segment in >sc1 that has the same name, saved weekly; used in case the system copy is damaged.

today.use_totals
: a statistical data base that describes month-to-date system resource availability and usage; prepared daily from meter_data and the PDTs.

usage_and_revenue.control
: a segment that defines the groups to be reported on by the usage_and_revenue command which is run daily under the crank.

usage_and_revenue.report
: a segment containing the output of the usage_and_revenue command; produced daily by the crank: suitable for printing.

URF
: the user registration multisegment file, giving names, addresses, and Person_ids; referenced when a person is registered to guarantee that his Person_id is unique at the site.

weekly.report
: a weekly summary of administrative operations.

yesterday.use_totals
: the previous day's copy of today.use_totals; used in preparing system.report.

Other data segments kept in this directory include backup copies of billing data from previous months and other temporary segments. The monthly bills and statistical reports are created in the >udd>SysAdmin>admin directory.

Processed log files are placed in the directory:

```
>udd>SysAdmin>admin>history
```

where they remain until they are deleted by the system administrator. They should be kept for about a month.

The SAT (in the >sc1>sat segment) and the PDTs (in the >sc1>pdts directory) in the >sc1 directory are copied daily (by the accounting job known as the "crank") into the directory:

```
>udd>SysAdmin>admin>safe_pdts
```

for both backup and accounting purposes. These copies are used daily, to produce month-to-date charge summaries, and, at the end of a billing period, to produce the bills and to reset the usage charges in PDTs stored in >sc1>pdt.

Copies of the usage totals and the projfile and reqfile segments for previous months are kept in another directory:

```
>udd>SysAdmin>admin>HF
```

They are kept here for some time after billing, so that the billing can be rerun if necessary. These segments should be dumped to tape and deleted after a reasonable period.

Contents of >udd>SysAdmin>lib Directory

Site-dependent system administration tools are kept in the directory:

```
>udd>SysAdmin>lib
```

This accounting library directory also contains:

biller.ec
: the exec_com segment invoked for billing.

err.ec
: the exec_com segment invoked when errors occur.

master.ec
: the exec_com segment that implements most of the accounting administrator commands.

starname_list
: a segment that lists the names to be counted by the disk_usage_stat program as it sweeps the system hierarchy accumulating disk statistics.

sys_admin_data
: a segment that contains registration parameters and an interlock to prevent multiple editing of the same segment.

sys_admin.value
: operated on by value_get and value_set (refer to *Multics Commands and Active Functions* manual, Order No. AG92).

util.ec
: the exec_com segment invoked by master.ec and biller.ec to perform utility functions such as deleting old copies of segments, if they exist, or dprinting copies of reports for a specified list of administrators.

All system administrators should use the add_search_rules command to add the >udd>SysAdmin>lib directory to their search rules as part of their start_up.ec. For information on the recommended access for each segment, refer to "Assuring the Security of the File System" later in this manual.

SECTION 13

MANAGING QUOTA

Storage system quota is a measure of how much total storage space ordinary segments (programs, data, text, etc.) occupy. Segments are made up of records with one record being equal to one page. All pages of a segment are of the same size and are 1024 words. If a segment is so subdivided it is said to be "paged." A segment always contains an integral number of records (i.e., 1,2 ...256).

Many directories have a set limit for the amount of quota they can use. This is called terminal quota. When the storage system tries to add a page to a segment in a directory which has terminal quota, the total number of records (pages) used in the directory is compared to the terminal quota for the directory. If the sum is greater than the terminal quota set, an error occurs because you are not allowed to go over the limit. This error is called Record_Quota_Overflow (RQO) and means that you have run out of quota for the directories.

Directories that do not have terminal quota draw their quota from the closest parent directory with terminal quota.

GIVING A PROJECT MORE QUOTA

Occasionally, a project becomes low on segment quota. To help you keep track of those projects that are running low on quota there is a daily disk report that lists all those projects that are approaching their quota limit. It is possible to write an automatic program that gives more quota to the projects in need and takes quota away from those projects that have extra quota. This is usually not done because the algorithms involved are complicated and the program still requires manual adjustment and intervention. It is better for you to encourage people to clean up their directories instead of continually increasing their quota when it runs out.

As the system administrator, you can decide which projects should be allotted more quota. After you initially set the quota on the root directory, you can move this quota down from the root directory to the >udd directory and down, level by level, to where it is needed.

SETTING QUOTA ON THE ROOT DIRECTORY

The set_quota command sets the total quota on the root directory, which stands alone at the top of the directory hierarchy. *You should only use the set_quota command on the root directory.* If you use the set_quota command on any other directory, significant problems can develop. Using the set_quota command on a directory other than the root can cause discrepancies in the disk_usage statistics compiled by the "crank". After you set the quota on the root directory, you can transfer quota to inferior directories using the move_quota command (described below). Use the set_quota command as follows:

```
set_quota > 50000
```

This command line sets quota on the root directory to allow 50,000 pages of storage.

MOVING STORAGE SYSTEM QUOTA FROM THE ROOT DIRECTORY

Once you have set the quota on the root directory you can move this newly created quota down from the root directory to the >udd directory using the move_quota command. You can then use subsequent move_quota commands to distribute this quota even lower to the project directory and then to an individual user's directory. To move the newly created quota from the root directory to the >udd directory, type:

```
move_quota >udd 1000
```

This command line moves 1000 records of new quota down from the root directory to the >udd directory.

To move this quota from the >udd directory to a project directory, type:

```
move_quota >udd>m 1000
```

This command line moves 1000 records from the >udd directory to the project directory named "m." To move this quota from the project directory to an individual user's directory, type:

```
move_quota >udd>m>Smith 1000
```

This command line moves 1000 records from the project directory >udd>m to Smith's directory.

MOVING QUOTA BETWEEN TWO DIRECTORIES

You can also move quota between two directories using the move_quota command or the priv_move_quota command. In order to move records of quota between two directories, one directory must be immediately inferior to (contained in) the other directory. To use the move_quota command, type:

```
mq >udd>m>Smith>sub1_dir 1000
```

This command line takes 1000 records from the quota on >udd>m>Smith and adds them to the quota on >udd>m>Smith>sub1_dir.

To take quota from the directory branch and move it back to the containing directory, use the move_quota command as follows:

```
mq >udd>m>Smith>sub1_dir>sub2_dir -50
```

This command line takes 50 records from the quota on >udd>m>Smith>sub1_dir>sub2_dir and adds them to the quota on >udd>m>Smith>sub1_dir. Note that the number of records is expressed as negative value.

You can also use the move_quota command to move quota into or out of an upgraded directory provided you are logged in at the authorization equal to the access class of the containing directory. An upgraded directory has an access class higher and more sensitive than that of its containing directory.

Another command that allows you to move quota into or out of an upgraded directory is the priv_move_quota command. You should use this command sparingly and only to fix problems. The directory system privileges are turned on while quota is being moved with this command. Use the priv_move_quota command as follows.

```
priv_move_quota >udd>m>Smith>upgraded_dir 1000
```

This command line moves 1000 records from >udd>m>Smith to >udd>Smith>upgraded_dir.

CLEANING UP DIRECTORIES AND SEGMENTS

An important part of managing quota is cleaning up directories and segments. This process is such an uninteresting duty it is often neglected. Use the delete (dl) command to delete segments that are no longer needed. To delete directories, use the delete_dir command.

There are a number of directories which are cleaned up on a daily basis. For example, the system continually adds entries to the syserr_log in the >sc1>syserr_log directory. The syserr log is a log of messages called syserr messages or hardcore messages, produced by the Multics supervisor and some of the online programs. Every day the crank deletes all entries that are older than N days (usually 10), using the date_deleter command. In this way, the size of the syserr_log directory remains relatively constant. For more information on syserr log, see Section 24 and Section 25.

QUOTA SALVAGING

Salvagers are a set of programs which do clean up work on the storage system. They detect damage and if possible, correct it. The quota salvager examines the disk storage in use to ensure that all quota used figures are accurate. The salvager attempts to rectify quota used inconsistencies.

The fix_quota_used command repairs inconsistencies in storage system quota used for a directory. Use the fix_quota_used command as follows:

```
fix_quota_used foo_dir
```

This command line repairs quota inconsistencies to the foo directory.

The normal use of the fix_quota_used command is from the fix_quota_used.ec or the x repair operator command (see below). When a quota (segment quota or directory quota) is found inconsistent and corrected, a message is printed. If the correction causes a directory to have greater quota used than allocated, another message is printed.

Use the x repair command to start (or stop) a multiprocess hierarchy repair. The repair can be an online directory salvage, a quota_used correction, or both. You can perform the repair over the entire hierarchy or any subtree. Up to 36 Salvager SysDaemon processes may participate in the repair, so that it is completed as rapidly as possible. To use the repair command, type:

```
x repair quota >udd 5
```

This command line performs quota used corrections beginning at >udd (the starting directory). Five Salvager SysDaemon processes participate in the repair.

Any of these operations automatically collate and sort all output, and dprint the results to salv_output.DATE.TIME and online_salvout.DATE.TIME, being salvager output and online error messages, respectively. The outputs are kept in the directory >udd>SysDaemon>Salvager and are automatically deleted when they become two weeks old.

INQUIRING ABOUT THE AMOUNT OF QUOTA

To inquire about the amount of quota, use the following commands:

- get_quota, gq
  This command returns information about the secondary storage quota and pages used by segments.

- get_dir_quota
  This command returns information about the directory quota and pages used by inferior directories.

- monitor_quota
  This command calculates storage of a directory and sends a warning message at the approach of a record_quota_overflow condition (this means that you are about to run out of space in the directory).

All of these commands are described below.

Get_Quota Command

To obtain information about secondary storage quota and pages used in segments, use the get_quota command. Status permission is required on each directory for which quota is desired.

The short form of output (the default case) prints the number of pages of quota used by the segments in the directory and the segments in any inferior directories that are charged against that quota. The output is prepared in tabular format, with a total, when more than one pathname is specified. When only one pathname is specified, a single line of output is printed.

The long form of output gives the quota and pages-used information provided in the short output. In addition, it prints the logical volume identifier of segments. Thus, a user can see what secondary storage charges the user's accounts are accumulating. If the user has inferior directories with nonzero quotas, it is necessary to print this product for all these directories to obtain the charge. Use the get_quota command as follows:

```
gq
```

This command line prints information about the working directory quota and pages used by inferior directories. Since no pathname is given, the working directory is assumed. The system prints:

```
quota = 0;  used = 2
```

Get Directory Quota Command

To obtain information about the directory quota and pages used by inferior directories, use the get_dir_quota command. Status permission is required on each directory for which quota is desired.

The short form of output (the default case) prints the number of pages of quota used by the segments in that directory and in any inferior directories that are charged against that quota. The output is prepared in tabular format, with a total, when more than one pathname is specified. When only one pathname is given, a single line of output is printed.

The long form of output gives the quota and pages-used information provided in the short output. In addition, it prints the logical volume identifier of segments, the time-record product in units of record days, and the date that this number was last updated. Thus, a user can see what secondary storage charges the user's accounts are accumulating. If the user has inferior directories with nonzero quotas, it is necessary to print this product for all these directories in order to obtain the charge. To use the get_dir_quota command, type:

```
get_dir_quota -lg
```

This command line returns quota information for the working directory. Since no pathname is given, the working directory is assumed. The long, -lg control argument includes the cumulative time_page product for the current accounting period. The system prints information about the directory quota and pages used by inferior directories:

```
quota:  0 pages (space is charged to superior directory)
used:  2 pages
remaining:  68 pages
sons volume:  Multics_Fed
```

Monitor Quota Command

Another command that helps you to keep track of the quota is the monitor_quota command. This command calculates storage of a directory and sends a warning message at the approach of a record_quota_overflow condition. It also sends a message when a record quota overflow condition actually occurs. You can use this command several times in a process to monitor several different directories. Use of the -off control argument stops monitoring of all directories. Use the monitor_quota command, as follows:

```
monitor_quota -warn DJones.Doc foo_dir
```

This command line sends the warning message regarding the foo directory to DJones in the Doc project. A limit of ten users can be listed with the use of the -warn control argument.

DIRECTORIES TO WATCH

You should be aware of the segments and directories that are not properly cleaned up. It is important to routinely clean up the directories listed below, because they have a tendency to accumulate many unnecessary segments.

1. >user_dir_dir>SysAdmin>admin>history (>udd>sa>a>history)

   The crank copies all the old log segments into this directory on a daily basis. This amounts to at least 300 pages a day. Since no accounting job ever deletes these segments, it is assumed that the site will do this "by hand". To eliminate this situation, you should add the date_deleter command (described in the *Multics Command and Active Functions* manual, Order No. AG92) to the crank to delete these segments weekly or to move them to some form of secondary storage.

2. >user_dir_dir>SysAdmin>admin>HF (>udd>sa>a>HF)

   This is a directory used by the monthly billing absentee that accumulates copies of different segments at the rate of about 150 pages per month. The hf directory is used by the billing absentee.

3. >reload_dir

   This directory usually contains a variety of old maps and error files that you should delete. A map is created in this directory whenever a reload of a hierarchy backup tape is done.

4. >dumps

   The copy_dump command copies dumps into this directory. Since a dump contains 2000 pages or more, a day or two of repeated crashes can quickly fill this directory. The initializer or other processes issue the copy_dump command so >dumps can exceed its quota. When this occurs (if >dumps is on the root logical volume), the root logical volume (rlv) fills up rather than causing a record_quota_overflow. To alleviate this problem, you should create a directory on another logical volume to contain dumps. After a dump is placed in >dumps, it should be moved to this new directory. When a process receives a fatal process error, copied dead processes are placed in >dumps>save_pdirs.

   This >dumps directory also contains copies of various process segments that are of little use to a nondevelopment site. These nondump segments should be deleted.

5. >system_control_1 (>sc1)

   This directory accumulates a number of small segments such as old copies of system tables and exec_coms that you should delete. Dumps of the answering service are named asdump.-. Date.Time. If the answering service gets repeated errors, it can generate asdumps until >sc1 runs out of quota. At that point the answering service continues to generate asdumps, each of which reports that there has been a record_quota_overflow in >sc1. The crank only cleans up the asdump if it is found in the >sc1 directory.

6. >udd>SysAdmin>admin (>udd>sa>a)

   This directory can contain old copies of some system tables that should be deleted.

7. >udd>Daemon>Volume_Dumper

   This directory often contains many old error files created by the volume dumper.

8. >udd>SysDaemon>Backup

   This directory often contains many old dump maps and error files. Since they are dprinted with -delete, special action by the system administrator usually isn't needed. However, if any maps are more than 5 days old, you should check to see if copies of the old maps were successfully dprinted and then delete the old maps manually. If the old maps were not dprinted, dprint the old maps again with -delete.

9. >udd>SysDaemon>Salvager

   This directory often contains many segments named online_salvout.date.time and salv_output.date.time. These segments are the output from the "x repair". The salvager deletes old copies of these segments, using the date_deleter command:

```
date_deleter -wd 14 salv_output.** online_salvout.**
```

   You should periodically inspect these directories for unnecessary segments.

10. >udd>SysDaemon>**
    >udd>Daemon>**

    You should periodically inspect these directories for unnecessary information.

11. >udd>SysAdmin>**
    >udd>SysMaint>**

    These directories most likely belong to people who do not have to pay for the disk storage. For this reason, you should assign a reasonable quota to these people.

MONITORING DISK SPACE

Monitoring disk space is an important part of controlling quota. To properly monitor disk space, the system administrator should ensure that the critical directories in Multics do not consume all of their quota.

You can also keep track of disk space using the list_vols command. This command allows you to keep track of the number of records that are left on a physical or logical volume.

The list_vols command prints information about the currently mounted physical or logical volumes. To find out how many records are left on the root logical volume, type:

```
list_vols_tt root
```

The system prints the following totals for the root logical volume.

Records	Left	VTOCEs	Left	PB/PD	LVName
50267	4026	20185	7874	pb	root

The control argument -tt does not print information for individual physical volumes but rather totals and prints information for each logical volume. For more information on the list_vols command, refer to the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Data migration is a way of forcing deletion of unused segments. Data migration means to move segments to offline storage if they haven't been referenced in the last N months, days, or weeks. A well designed data migration system can provide the means for listing and retrieving segments that have been migrated. Use the disk_usage_stat command (documented in *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64) to analyze how much storage space you can gain by using a data migration system.

Additionally, you can use the date_deleter command to delete old segments. If a segment has not been modified in the last N days, this command deletes that segment. Use this command with caution because it will delete segments that have not been used recently. For more information on the date_deleter command, refer to the *Multics Commands and Active Functions* manual, Order No. AG92.

Finally, it is always a good idea to assign one person who is responsible for monitoring disk space on a daily basis. This person would ensure that there is adequate space available in the critical directories and on all the logical volumes.

PART V
MANAGING STORAGE SYSTEM VOLUMES

SECTION 14

DISK VOLUME REGISTRATION

Disk volume registration requires an understanding of physical and logical volumes. This section contains information on initializing a physical volume and modifying and deleting a volume registration. The procedures required to obtain volume registration information are also described later in this section.

UNDERSTANDING PHYSICAL AND LOGICAL VOLUMES

Disk packs are the principal means of storing information on Multics. A hardware disk pack is considered a physical volume. The Multics system permits the grouping of physical volumes into sets, with each set considered a single unit. The group of physical volumes joined together in the set are considered a single logical volume.

When a logical volume is created a registration record is created for it. This registration record contains the following information:

- a list of the physical disk volumes comprising the logical volume.
- the logical volume identifier (name and/or uid).
- public or private switch
- the list of directory identifiers (name and/or uid) defining all directories that would be legitimate to put on this particular logical volume.

Each physical volume retains the following attributes:

- all pages of a given segment are allocated in the same physical volume.
- a physical volume always has its own volume map (an array containing one entry for each physical 1024 - word record of the volume).
- a physical volume always has its own Volume Table of Contents (VTOC). This is an array containing one entry for each segment or directory stored in the volume.
- a physical volume always has its own registration record; the registration record is addressable by a unique name or unique id associated with the physical volume.

There are also two special volumes known as the root physical volume (RPV) and the root logical volume (RLV).

The RPV is the single physical volume required to bring Multics up as far as ring 1 command level, and do ring 1 functions like reloading volumes and performing some kind of crash recovery. The RPV also contains the root directory, ">", as one of its segments. The RLV is the only logical volume that contains directory segments.

REGISTERING A PHYSICAL OR LOGICAL VOLUME

You must register a physical volume before you initialize a physical volume. To register a new physical volume, use the add_volume_registration (avr) command. If the logical volume does not already exist, this command also registers the new logical volume. For a complete description of this command, refer to the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

INITIALIZING A PHYSICAL VOLUME

To initialize a physical volume, use the init_vol initializer command. This command writes the label of a new physical volume and sets up its Volume Table of Contents-VTOC (an array containing one entry for each segment stored in the volume). This command also sets up the volume map (an array containing one entry for each physical 1024 word record of the volume) of a new physical volume. About 5% of each volume is used for the VTOC and volume map. This operation destroys any previous contents of the physical volume. The init_vol command queries the operator before destroying the label of any pack that appears to be a validly labeled pack. A message giving the pack's physical volume name and time of last use is displayed. For further information on this command refer to *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

MODIFYING A VOLUME REGISTRATION

To modify a logical or physical volume registration, use the change_volume_registration command. You should only change the name or unique ID attributes of a physical or logical volume to correct a registration file that is inaccurate. For a complete description of this command, refer to the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

DELETING A VOLUME REGISTRATION

To unregister a logical or physical volume, use the delete_volume_registration command. Unregistering a volume and then re-registering it is usually an error. For a complete description of this command, refer to the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

OBTAINING INFORMATION ABOUT VOLUME REGISTRATION

To obtain registration information about logical or physical volumes, use the list_volume_registration command. To list registration information for a physical volume, type:

```
list_vol_registration pv volume_name
```

To list registration information for a logical volume, type:

```
list_vol_registration lv lv_name {control_arg}
```

Use the list_vol_registration command as follows:

```
lvr lv libraries
```

This command line returns registration information about the logical volume named "libraries".

The system responds with:

```
lvname:     libraries
lvid:               522715230333 (!hHmhJFWbBBBBBB)
public:             yes
owner:              Initializer.SysDaemon
min_access_class:   0:000000 ()
max_access_class:   7:777777 (system_high)
acs_path:           >lv>libraries.acs

npv:                2

  pvname: ldd0
  pvid:             522715756711 (!hHmkpkGbBBBBBB)
  serial:           registered by crash recovery
  model:            501
  location:         online
  date registered:  12/16/84 0051.9 est Sun

  pvname: ldd1
  pvid:             523045104777 (!hJGgcKzbBBBBBB)
  serial:           registered by crash recovery
  model:            501
  location:         online
  date registered:  12/16/84 0052.1 est Sun
```

Using the control argument brief (bf) when lv is specified, lists only physical volume names.

For example:

```
lvr lv PubA -bf
```

This command line lists the physical volumes associated with the logical volume "PubA". The system responds with:

```
PubA_01
PubA_02
```

where "PubA_01" and "PubA_02" are the physical volume names.

For a complete description of the list_volume_registration command, refer to the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

SECTION 15

MANAGING LOGICAL VOLUME ACCESS

This section describes the access you need to administer a volume. The procedure for finding and setting access to a logical volume access control segment (ACS) is described below.

UNDERSTANDING ACCESS NEEDED TO ADMINISTER A VOLUME

Each logical volume has an ACS associated with it. The Multics access control system provides for control of various resources via the use of ACSs. An ACS is normally a zero-length segment associated with a system resource.

The effective access mode of a user process to an ACS determines the access of that process to the resource identified with it. The entryname of an ACS generally describes the system resource it is associated with. For example, the ACS for tape drive 6 has the entryname tape_06.acs. All ACSs have the suffix acs. The system checks the user's access to the appropriate ACS when his process requests use of a controlled system resource. For more information on ACS refer to the *Multics Programmer's Reference Manual*, Order No. AG91.

You must have rw (read, write) access to the ACS associated with a logical volume to attach the volume and use any segments on the volume. In addition, if you have e (executive) access to the logical volume, you can manipulate the quota specifications for subtrees on the logical volume and create master directories on the logical volume.

FINDING LOGICAL VOLUME ACCESS CONTROL SEGMENTS

To find the ACS segment associated with a logical volume, use the list_volume_registration command as follows:

```
lvr lv libraries
```

where "lvr" is the short form of the list volume registration command, "lv" is the literal string, and "libraries" is the logical volume name. The system responds with information about the logical volume names "libraries". For more information on this topic refer to "Obtaining Information About Volume Registration" in this manual.

SETTING ACCESS TO A LOGICAL VOLUME ACS

There is an empty ACS for each logical volume known to the system. The access control list (ACL) for each segment defines the permissions to use the respective logical volume. The pathname for the ACS of the logical volume is located in the volume registration and can be displayed with the lvr command.

The ACL for each segment defines the user's access rights to use the respective logical volume. To have executive access to a logical volume, a user must have e access to the ACS for that volume. Executive access is required to perform the following functions on a given logical volume.

- set, change, or delete volume quota
- set the owner of a logical volume
- set quota on a master directory on a logical volume
- list all master directories on a logical volume

In the absence of an ACS, the default access is rew to the owner of the logical volume. For all other users, the default access is rw if the volume is public and null if the volume is private.

To set access to a logical volume, use the set_acl command. This command changes the ACLs of the ACS segment. To create an ACS, use the create command. To associate an ACS pathname with a new logical volume, use the add_volume_registration (avr) command. To change the ACS pathname, use the change_volume_registration command.

SECTION 16

USING A LOGICAL VOLUME

Logical volumes can be either public or private and they can be mounted or demounted with the add_lv or the del_lv request respectively. You can move segments to another logical volume, create a quota account on a logical volume or change the quota available in a quota account using the commands described below.

ORGANIZING DISK STORAGE INTO LOGICAL VOLUMES

The Root Logical Volume (RLV) contains the root directory. The RLV must always be mounted because the information contained in the RLV must always be available to the system.

MOUNTING AND DEMOUNTING LOGICAL AND PHYSICAL VOLUMES

When you issue an add_lv or del_lv command, you must specify the logical volume for which you are requesting the operation. The system examines the registration record of the specified logical volume and then determines which physical volumes are members of the logical volume.

The operator can only mount a logical volume if there are enough disk drives available to accommodate all physical volumes included in the logical volume. To mount a physical volume is to physically place it on a drive (if demountable) and cycle up that drive. This action is performed by the operator, not by the software.

A logical volume is mounted when all of the physical volumes it contains are mounted and accepted, and calls have been made to the supervisor to establish the presence of this complete logical volume on line.

You can only demount a logical volume after all of its physical volumes have been updated with the current information residing in the memory.

To demount a logical volume use the del_lv command (see *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64). This makes the contents of the logical volume unavailable to users. You cannot demount a logical volume if any of the following cases occur:

- if the logical volume is the root logical volume
- if any of the logical volume's physical volumes contain a partition designated by a PART configuration card
- if the logical volume has been used to store process directory segments with the set_pdir_volumes command or the add_pdir_volumes command.

NOTE: If you have used the set_pdir_volume or add_pdir_volume commands you can use the vacate_pdir_volume to force per process segments (segments in process directories) off the specified logical volumes. These segments are spread evenly over the remaining volumes in the PDIR volume set. Once this operation is complete, it is possible to delete the logical volumes that have been vacated. For more information on these commands, refer to the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

DEFINING PUBLIC AND PRIVATE VOLUMES

The two types of logical volumes used on the storage system are public and private. When the logical volume is registered with the add_vol_registration command the distinction is made between public and private. (See "Disk Volume Registration" in this manual.)

If a logical volume is *public*, access to it is unrestricted, and its physical volumes are usually permanently mounted. It is not necessary for a public logical volume to be attached to your process explicitly with the attach_lv command.

If a logical volume is *private*, access to it is restricted, and its physical volumes are usually not permanently mounted. To use a private logical volume, the user must request that the logical volume be mounted. When a segment is stored on a private logical volume, a process must attach the volume (via the attach_lv command) before it can use the segment. The volume remains attached until the process explicitly detaches it (via the detach_lv command).

CREATING CONVENTIONS FOR DISK VOLUMES NAMES

A disk volume name can contain up to 32 alphanumeric characters and may not have any spaces. To eliminate problems, it is important to use a consistent naming convention. For example, if you have a logical volume named "PubA" and you want to add physical volumes to "PubA", you should continue to use the name "PubA" at the beginning of the name of each new physical volume that you add. In the following example the logical volume PubA contains three physical volumes named PubA_01, PubA_02 and PubA_03:

```
Logical Volume    PubA
Physical Volume   PubA_01
Physical Volume   PubA_02
Physical Volume   PubA_03
```

Using this type of naming convention enables you to easily locate the physical volumes that reside on a particular logical volume.

ALLOCATING LOGICAL VOLUME QUOTA

You should ensure that a logical volume does not consume all of its quota. When a logical volume begins to reach its quota limit, there are several actions you can take, as described below.

Regaining Space

Use the sweep_pv to regain space lost to connection-failed segments. This should not regain much space because a site should be doing this routinely for all disks every week or two. You can also use the sweep_pv command to perform utility functions that require walking through the VTOC of a mounted physical volume. These functions include listing the contents and unconnected VTOCEs, deleting these VTOCEs, and evaluating physical volumes (for logical volume compression). Use the sweep_pv command as follows:

```
sweep_pv subA -list
```

where "sweep_pv" is the name of the command, "subA" is the name of the physical volume, and "-list" is the control argument that creates a listing file containing the VTOC index, time listed, and pathname of every segment on the volume at the time its VTOCE is scanned.

Adding a Physical Volume

You can expand the logical volume by adding another physical volume (disk pack). You need a spare drive to perform this function. To expand a logical volume follow the steps listed below:

1. To format the disk pack, you must use MTR for the MSU 500 and 501 packs. For the MSU 451 pack, you can use either MTR or the BOS FMT with the format_disk_pack command. It is recommended that you use MTR with the MSU 451 because MTR is faster and you can use it while the system is running.

2. If a drive must be added to the configuration, and the configuration deck is changed, the disk table may have to be re-created at the next bootload. Use the list_disk command to make a list of all the disk packs. A copy of this list should be kept by the console to make it easier to re-create the disk table. A site can always add a mechanism for storing away these various disks and assignments and then you can run an "x" command to add them all back. If the disk table has to be re-created, it will be necessary to perform add_vol initializer commands for each old volume.

3. To add the new physical volume to the logical volume specified by the -lv control arg, issue an add_vol_registration privileged command with the -pv control argument.

4. To write the label and VTOC of the new physical volume , issue an init_vol initializer command. If there are partitions, defective track space, or special space requirements, use the -special control argument.

5. To begin using the new physical volume mounted on the specified disk drive, issue an add_vol initializer command.

6. If dumping by physical volume name, add the volume name to the volume dumper control files.

If an existing logical volume is expanded, all new segments created on that logical volume are placed on the new physical volume for as long as the new volume has the least space. This policy might lead to excessive arm motion on the drive of the new volume and degraded performance. Use the sweep_pv command and the inhibit_pv command to move segments between packs or inhibit segment creation on any pack.

Moving Segments to Another Logical Volume

One of the options for making space on a logical volume is to move an entire subtree. To move an entire subtree use the steps listed below:

1. Make a dump of the original subtree with the "backup_dump" command (see *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64).

2. Create a new master directory on the new logical volume.

3. Use the retrieve command (see *Multics Commands and Active Functions* manual, Order No. AG92) to retrieve the subtree dumped to tape into the new master directory.

4. Delete the original (old) subtree.

Finding and deleting unnecessary segments is the best way to gain space because it truly gains space instead of just shuffling it around. Finding unnecessary segments is discussed under "Managing Quota" in this manual.

MANAGING SPACE ON THE VARIOUS LOGICAL VOLUMES

The following paragraphs discuss the various types of logical volumes.

Private logical volumes
These disks are solely the responsibility of the volume executive and are not needed for correct running of system.

Public logical volumes (not used for process directory segments)
A public logical volume is a storage system logical volume to which access is unrestricted. The list_vols command marks these volumes with "pb". Use the list_vols command to see how much space is left on the disk volumes. These volumes should not become more than 90% full.

Public logical volumes (used for process directories)
The list_vols command marks these volumes with "pb pd". These volumes are important, because they contain process directories that might belong to an important system process. These volumes should not become more than 85% full.

Root Logical Volume (RLV)

The RLV contains all of the files and directories needed to bring Multics up all the way to ring 4. This logical volume is the most critical, because a full root logical volume could crash the system. It is also susceptible to gathering a lot of unnecessary directories. You should not allow the RLV to become more than 75% full.

CREATING A QUOTA ACCOUNT ON A LOGICAL VOLUME

To create a quota account on a logical volume, use the set_volume_quota (svq) command described below. This command sets a user's account on a logical volume.

The volume does not have to be mounted to use the svq command, but the user must have executive (e) access to the logical volume (see "Managing Logical Volume Access" in this manual). If the volume quota is set less than the quota account's current quota used, the quota is changed as directed, but a warning message is printed.

The command is frequently used to set up a quota account for someone else and you actually use the account pack. Use the (svq) command as follows:

```
svq foo 1000 *.Pubs
```

This command line set the "Pubs" project to 1000 records and allows anyone on the "Pubs" project to use up to 1000 pages of quota.

Change is the amount of quota, or the amount of quota change, and is specified as follows:

```
+n adds n records to the quota
-n subtracts n records from the quota
 n sets the quota to n records
```

CHANGING THE QUOTA AVAILABLE IN A QUOTA ACCOUNT

The system administrator can increase or decrease the number of records in a quota account by using the set_volume_quota command described below. To use this command, the user must have "e" access to the logical volume. It is not necessary that the volume be mounted. If the volume quota is set less than the quota account's current quota used, the quota is changed as directed, but a warning message is printed. You can add or subtract records from the quota by using +n or -n, respectively. To increase the quota on a logical volume, use the set_volume_quota command with +n as follows:

```
svq foo +1000 McElvenny.Pubs
```

This command line increases the number of records in Smith's quota account is increased by 1000 (+1000) records.

To decrease the quota on a logical volume, use the set_volume_quota command with -n as follows:

```
svq foo -1000 McElvenny.Pubs
```

This command line decreases the number of records in McElvenny's quota account by 1000 (-1000) records. If an account is not given, the total quota of the logical volume is changed.

DELETING A QUOTA ACCOUNT

If you are an owner or manager of a logical volume, you can delete a quota account for a logical volume. You must have execute access to the logical volume to do this. The quota account cannot be deleted if there are still master directories whose quotas are charged against the account to be deleted. Such directories must be deleted or transferred to another account using the set_mdir_account command. To delete a quota account for a logical volume, use the delete_logical_volume_quota (dlvq) as follows:

```
dlvq foo McElvenny.Pubs
```

This command line deletes the quota account on the logical volume "foo". "McElvenny.Pubs" is the Person_id.Project_id of the user with a quota account for the logical volume.

SECTION 17

USING A QUOTA ACCOUNT

As a system administrator, there a number of functions that you can perform with a quota account. These functions are described below.

CREATING A MASTER DIRECTORY

When you create a new directory its segments (by default) reside on the same logical volume as the segments of its parent directory. If you want the segments in the new directory to reside on a different logical volume, you must create a *master directory*. A master directory is simply the point where the hierarchy "branches out" to another logical volume. A master directory has its own quota limit and does not receive a quota limit from its parent. Figure 17-1 illustrates the relationship among directories, master directories, and logical volumes.

To create a master directory on a logical volume, you must have a quota account on the logical volume and executive access to the logical volume. This account must contain enough quota to satisfy the request. A master directory must always have a nonzero quota; therefore, the -quota control argument must always be given when creating a master directory. The quota on a master directory can never be set less than the current number of records being charged against the master directory. You can create a master directory even though the logical volume is not mounted. Use the create_dir command with the -logical_volume control argument as follows:

```
cd subB -lv volz -quota 1000
```

This command line creates a new master directory named "subB" and is given a quota of 1000 records. The name of the logical volume containing the segments is "volz".

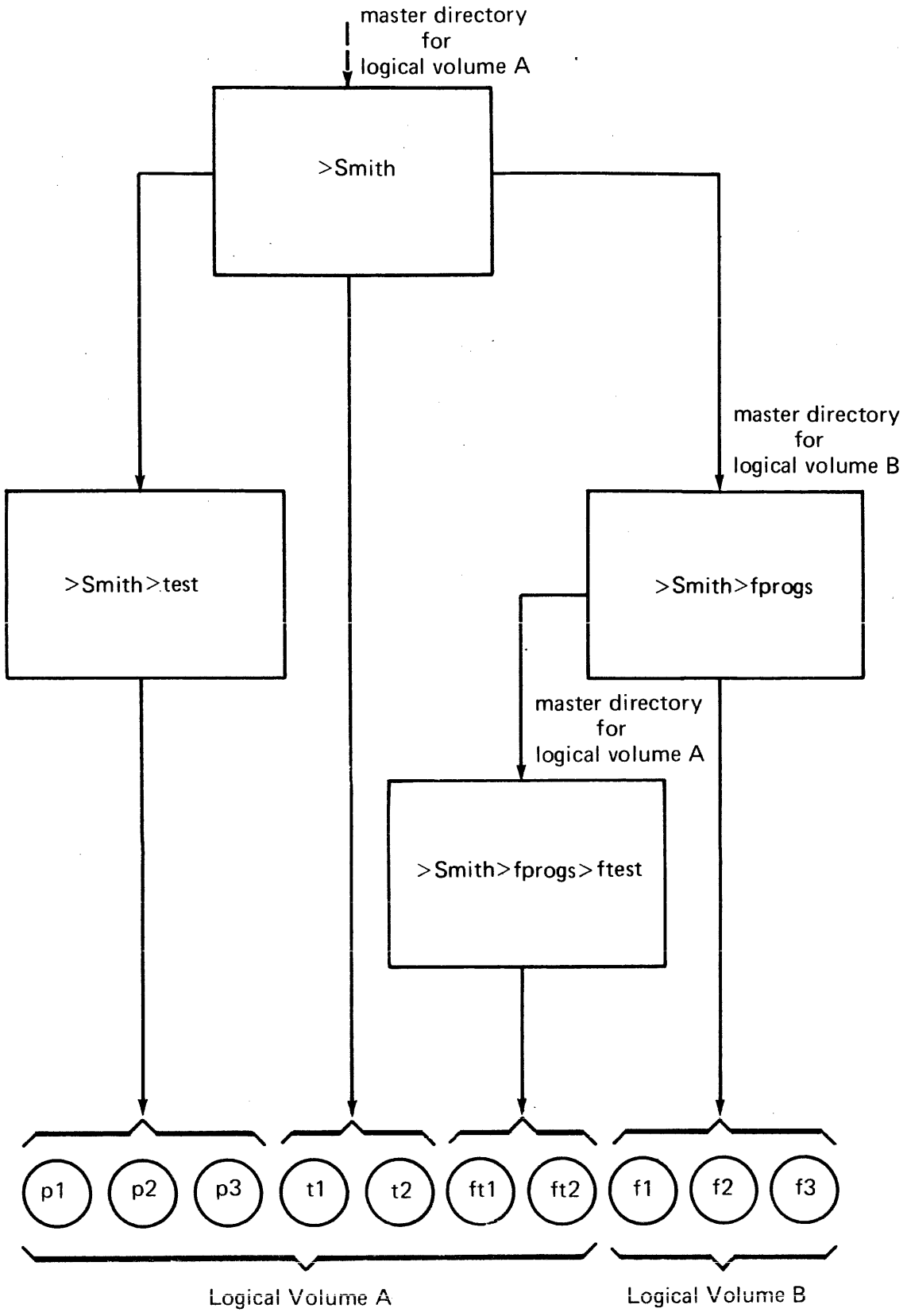

Figure 17-1. Relationship of Directories to Logical Volumes

CHANGING THE OWNER OF A MASTER DIRECTORY

To use the set_master_directory_owner command you must have "e" access on the logical volume containing the master directory. The volume need not be mounted. To change the owner of a master directory, use the smdo command as follows:

```
smdo subB DJones.Doc
```

where "smdo" is the short form of the set_master_directory_owner command, "subB" is the pathname of the master directory, and "DJones" (from Doc project) is the name of the new owner of the master directory.

DELETING A MASTER DIRECTORY

If you find that you no longer need a master directory you can use the delete_dir command described below to delete the directory. The delete_dir command causes the specified directory and any entries it contains to be deleted. All inferior directories and their contents are also deleted. If the -force control argument is not supplied, delete_dir asks you whether to delete the specified directory; it is only deleted if you type "yes" in answer to the query. When deleting a nonempty master directory, or a directory containing inferior nonempty master directories, you must have previously mounted the logical volume(s). Use the delete_dir command as follows:

```
dd subB -fc
```

where "dd" is the short form of the delete directory command and "subB" is the master directory that is deleted. The "-fc" control argument deletes the directory without issuing a query.

CHANGING THE QUOTA ON A MASTER DIRECTORY

A master directory has its own quota limit and does not receive a quota limit from its parent. You can change or increase the quota on the master directory using the set_master_dir_quota command. You must have modify permission on the master directory and must be the owner of the master directory, be a volume administrator, or have the same quota account as the master directory.

If the quota is being increased, the master directory's quota account must have sufficient volume quota to satisfy the request.

To change the quota on a master directory, use the set_master_dir_quota command as follows:

```
smdq subB 1000
```

where "smdq" is the short form of the set_master_dir_quota command and "subB" is the pathname of the master directory. This command line changes the quota to 1000 records.

OBTAINING QUOTA FROM A NEW ACCOUNT

To obtain quota from a new account, use the set_mdir_account (smda) described below. You must have "e" access on the logical volume containing the master directory. The volume need not be mounted.

The quota for the master directory is returned to the old quota account and is withdrawn from the new quota account, which must have sufficient quota to allow this. To set the quota account of a master directory, use the smda command as follows:

```
smda subB DJones.Doc
```

This command line changes the quota account on the master directory "subB". "DJones" (from the Doc Project) is the name of the new quota account of the master directory.

LIST MASTER DIRECTORY COMMAND

To print the logical volume and master directory quota, use the list_mdir command. The volume need not be mounted to use this command. You must have "e" access to the logical volume to use the -owner, -account, and -all control arguments. The -quota control argument prints only quota information and the -directory (-dr) control argument prints only master directory information. To print the logical volume and master directory, type:

```
lmd work
```

where "lmd" is the short form of the list_mdir command and "work" is the name of the logical volume.

The system responds with:

```
QUOTA PATHNAME
100 >udd>Fed>Jones>sub
250 >udd>Fed>Jones>sub1>sub2
350 records of quota assigned, 500 available.
```

To obtain information about all users of a logical volume, use the -all control argument with the list_mdir command. To use the -all control argument, you must have "e" access on the logical volume. Use the list_mdir command as follows:

```
lmd libraries -all
```

The system responds with:

```
ACCOUNT          QUOTA       PATHNAME
*.SysMaint       1400        >system_library_obsolete

ACCOUNT                      ASSIGNED    AVAILABLE
Initializer.SysDaemon        0           0
*.SysMaint                   1400        10000
```

PART VI
ASSURING SYSTEM SECURITY

SECTION 18

ASSURING THE SECURITY OF THE FILE SYSTEM

There are three types of access control on Multics that ensure the security of the data stored on the system:

- Discretionary access control – allows users to grant or deny other users access to their segments, directories, and resources at their own discretion by using Access Control Lists (ACLs)
- Nondiscretionary access control – enforces administrative policies concerning access to information with the Access Isolation Mechanism (AIM)
- Intraprocess access control (ring mechanism) – provides the ability for programs to enforce arbitrary access control policies that go beyond ACL and AIM controls using the Multics ring mechanism.

A complete description of each of the above types of access control is discussed in detail in the *Multics Programmers' Reference Manual*, Order No. AG91.

Multics has many system data bases that are potentially accessible to users. You must decide whether to grant access to these data bases to some, none, or all of the users. Granting users to some of these data bases allows users to learn about system overhead information like meters. In other cases, granting access allows users to find out about other users activity. In the sections to follow, all of the important data bases are listed.

ACCESS TO SPECIAL PROJECTS AND USERS

Different users of your system need different access to system data. To make it easier to administer access control, you should grant access to system data to projects rather than individuals. This allows you to determine who has special access on your system by checking the users registered on the projects instead of surveying the ACL's of all of the different system data segment, directories, and gates. This has the added advantage of encouraging administrators to avoid logging in with special access when they do not need it. A user with privileged access must be very careful to avoid mistakes (and Trojan horses), and avoid running untrusted programs. Thus, it is better if users always have accounts on projects with no special access for use when they do not need special access. However, under some circumstances like controlling access to send admin commands to the Initializer process, it is better for accountability purposes to give access by individual userids.

The standard Multics accounting start_up defines the following projects with special access:

- SysAdmin
  - system administrators
  - system security administrators

- SysMaint
  - system maintenance
- SysDaemon
  - special system processes
- Daemon
  - less privileged processes than those on the SysDaemon project
- HFED
  - Honeywell Customer Service Representatives
- Operator
  - operators

The remaining projects are ordinarily normal users of the system.

System Administration Personnel

As system administrator, you have access to the entire file system but are still controlled by AIM restrictions. Only system security administrators should have access to circumvent AIM restrictions.

System Security Administrators

System security administrators are distinguished by their ability to circumvent AIM. The capability to circumvent AIM is granted by access to the system_privilege_ gate (see Section 23). Access to the system_privilege_ gate distinguishes system administrators from system security administrators.

The Multics standard accounting start up assumes that system security administrators log in on the SysAdmin project. Since they are given access to the gates that circumvent AIM as individuals, they are an exception to the general principle above about granting access to projects. Since there should be a vary small number of them, this is usually acceptable. You may choose, however, to create a separate project for system security administrators. You might call it SysSec.

System Maintenance Personnel

System maintenance personnel belong to the SysMaint project and have access to everything on the system except system administration data bases (for a list of administrative directories, see "Access on System Administration Directories"). However, system maintenance personnel have the ability to force access on these directories, if required, by using the sac command (see the *Administration, Maintenance and Operations Commands* manual, Order No. GB64). System maintenance personnel can, indirectly, violate AIM restrictions since they have access to the hphcs_ gate. However, the system_privilege_ gate permits direct circumvention of AIM restriction and access to this gate should not be given to system maintenance personnel.

Members of the SysMaint project (system maintainers) fix system problems, view dumps, and perform other system maintenance procedures. The SysMaint project should not be given access to perform registration and accounting.

The SysMaint project directory is >udd>SysMaint or >udd>sm.

System Operators

System operators log into the system in two ways:

- as interactive users on normal terminals
- as operators on initializer terminals and the console.

When logged in as normal users, they access the system as ordinary users. Granting access to the Operator project gives them access to do their job. You should grant them access to see the status of the system as explained below. You may want to grant them access to send admin commands, send daemon commands, or perform other maintenance operations. This depends on your site policy.

When logged in at initializer terminals or the console, operators can only enter initializer commands. For most of these commands, no access control checks are made. However, you can restrict the ability of individual operators to manipulate particular daemons.

Ordinary Users

Ordinary users are those who belong to projects other than SysAdmin, SysMaint, SysDaemon, and Daemon. Among this group are included the field engineers who belong the the HFED project. Members of there HFED project must have access to a series of privileged operations.

Access rights for the general user population are based on site policy. This can be determined by classifying the information as:

- information that it is everybody's business to see
- information that is everybody's business to see if you are running an "open" site
- information that some users should see
- information that no member of the general user population should see

The recommended access that should be given to each class of user on the system for specific system files and tables is given below.

ACCESS ON LIBRARY DIRECTORIES

The following library directories contain the standard Multics command and subroutine software:

- >system_library_standard - contains most system commands and subroutines
- >system_library_1 - contains a large set of subroutines that are reloaded each time the system is booted, many supervisory programs, and software needed to perform system reloads
- >system_library_tools - contains software for system programmers
- >system_library_unbundled - contains unbundled software
- >system_library_inst_maint - installation maintained

The table below details the recommended access to the library directories.

Table 18-1. Library Directory Access

Library Directory	Personid	Access
>system_library_standard >system_library_tools >system_library_unbundled	*.SysDaemon.* *.SysMaint.* *.SysAdmin.* *.*.*	sma sma sma s
>system_library_1 (automatically set by the system)	Initializer.SysDaemon *.*.*	sma s

To set the proper access to system_library_tools for users belonging to the SysMaint project, type:

```
sac sa >system_library_tools sma *.SysMaint.*
```

For more detailed information on how to set ACLs for users, see the *Multics Programmer's Reference* manual, or type "help sa".

ACCESS ON SYSTEM ADMINISTRATION DIRECTORIES

You can configure a Multics site to be very secure, or moderately secure. A very secure Multics site would not allow access to any system-maintained information other than that absolutely required by a project and the users associated with the project. A moderately secure site would allow users read access to noncritical files for perusal and general instruction.

For the purposes of this section, all access recommendations are based on a moderately secure Multics site: read access is given to all users for all noncritical files.

The Multics system administration software (and other system software) depends on the existence of certain directories. These directories, listed below, are created and initialized at a new Multics site by the acct_start_up.ec segment.

- >system_control_1 or >sc1
- >udd>SysAdmin>admin
- >udd>SysAdmin>lib
- >daemon_dir_dir
- >lv
- >site
- >sc1>admin_acs
- >sc1>rcp
- >sc1>as_logs
- >sc1>syserr_log
- >sc1>mc_acs

The recommended access for all of the directories listed above is shown below.

```
sma       *.SysDaemon
sma       *.SysAdmin
sma       *.SysMaint
s         *
```

ACCESS IN THE DIRECTORY > sc1

The contents of the >sc1 directory include the special segments listed below. The recommended access for each segment is also specified below the segment name.

There are several segments in >sc1 whose ACLs are set automatically. They are:

- as_request.ms
- sat
- >sc1>pdt>project_name.pdt

Note that each time an individual is assigned project administrator status, they are automatically assigned read access to the corresponding PDT even if they already have rw access. *Whenever a system administrator is also a project administrator, it is important to change the ACL term to rw.*

as_request.ms

adros	Initializer.SysDaemon.*	a message segment used by all system users to communicate with the answering service to perform various system functions. Note that access to as_request.ms is created automatically.
ao	**.*	

absentee_user_table

rw	Initializer.SysDaemon.*	a table that has an entry for each absentee process, with the name, accounting data, process identification, and arguments for each absentee job that is running.
r	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.Operator.*	
r	*.SysMaint.*	
r	Data_Management.Daemon.*	

absentee_foreground.ms

adros	*.SysDaemon.*	the foreground absentee queue (a ring 1 message segment).
adros	*.SysAdmin.*	
adros	*.SysMaint.*	
aros	*.*.*	

absentee_N.ms

adros	*.SysDaemon.*	background absentee queue N (a ring 1 message segment).
adros	*.SysAdmin.*	
adros	*.SysMaint	
aos	*.*.*	

admin.ec

rw	Initializer.SysDaemon.*	an exec_com segment containing sequences of commands that the operator can execute by using the exec command (described in the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64).
r	Card_Input.Daemon.*	
r	*.SysDaemon.*	
rw	*.SysAdmin.*	
rw	*.SysMaint.*	
r	*.HFED	
r	*.Daemon	

answer_table

rw	Initializer.SysDaemon.z	a table that has an entry for each interactive process, with the name, accounting data, and process identification of the logged-in user.
r	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.Operator.*	
r	*.SysMaint.*	
null	*.*.*	

as_logs>log

rw	Initializer.SysDaemon.*	a log of events of interest to the initializer; many initializer messages typed to the operator are
r	*.SysDaemon.*	

r	*.SysAdmin.*	also recorded in this segment.
r	*.SysMaint.*	

cdt

rw	Initializer.SysDaemon.*	the channel definition table (CDT), containing an entry for each terminal channel, specifying per-channel attributes.
r	*.Operator.*	
r	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.SysMaint.*	

default_system_search_rules

r	*.*.*	a text segment containing the default search rules for users on the system.
rw	*.SysAdmin.*	
rw	*.SysMaint.*	

daemon_user_table

rw	Initializer.SysDaemon.*	a table that has an entry for each daemon process, with name, accounting data, process identification, and source identifier for each daemon entry currently logged in.
r	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.SysMaint.*	
r	*.Operator.*	
r	Data_Management.Daemon.*	
null	*.*.*	

fnp_crash_notify.ec

rw	*.SysMaint.*	an exec_com involved whenever a FNT crashes.
rw	*.SysAdmin.*	
r	*.SysDaemon.*	
r	*	

installation_parms

rw	Initializer.SysDaemon.*	a table containing site-settable parameters, including shift definitions and prices.
rw	*.SysDaemon.*	
rw	*.SysAdmin.*	
r	*.SysMaint.*	
r	*.*.*	

login_help

r	*.*.*	a segment that is typed if a user types "help" instead of "login".
rw	*.SysAdmin.*	
rw	*.SysMaint.*	

mc_anstbl

rw	Initializer.SysDaemon.*	the message coordinator's table of terminal channels.
r	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.SysMaint.*	

message_of_the_day

rw	Initializer.SysDaemon.*	a segment containing messages addressed to all users: edited by the operator command, message, or any text editor; printed either by the default process overseer at log-in time or by the commands, "help motd" or print_motd.
rw	*.SysDaemon.*	
rw	*.SysAdmin.*	
rw	*.Operator.*	
rw	*.SysMaint.*	
r	*.*.*	

mgt

rw	Initializer.SysDaemon.*	the master group table (MGT) specifying the guaranteed load units for each load control group and listing the current occupancy of all groups. It is accessed at each login to determine whether the attempted login would overload the system; it also contains the minimum percentage of system resources to be given to each work class and determines the work class membership of each load control group.
rw	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.SysMaint.*	

MRT

rw	Initializer.SysDaemon.*	the message coordinator's message-routing table.
r	*.SysDaemon.*	
r	*.SysMaint.*	
r	*.SysAdmin.*	

opr_query_data

rw	*.SysAdmin.*	A segment used for communication between the CSR'S running T&D and the operators.
rw	*.SysMaint.*	
rw	*.HFED	
rw	Initializer.SysDaemon	

PNT

rw	Initializer.SysDaemon.*	the person name table (PNT), containing the Person_ids of all registered persons; referenced at all logins (except those of anonymous users) to determine the user's password; referenced at login to check for a default Project_id, person_authorization, and password generation flag; and also referenced at card input time by the card input process, including RJE process. Note that, while portions of the user entry in the PNT are stored in encrypted form, any encryption algorithm is susceptible to a sophisticated, computer-assisted code-breaking effort. Therefore the System Administrator should ensure that access to the PNT is as restricted as possible. In general, only the
r	IMFT.Daemon.*	
rw	*.SysDaemon.*	
rw	*.SysAdmin.*	
r	Card_Input.Daemon	

SysAdmin and SysDaemon projects should have access to the PNT.

rate_structure_N

rw	Initializer.SysDaemon.*	where N is a digit in the range 0 through 9. The segment rate_structure_N contains the pricing information for the Nth rate structure defined in installation_parms. By default, a site uses only a single rate structure, number 0. This rate structure is contained in installation_parms, and therefore, the name rate_structure_0 is an additional name on the segment installation_parms. If the site defines new rate structures, additional rate_structure_N segments are created to contain the pricing information.
rw	*.SysDaemon.*	
rw	*.SysAdmin.*	
r	*.SysMaint.*	
r	*.*.*	

reconnect.ec

rw	*.SysAdmin.*	the default exec_com run when a disconnected process reconnects.
rw	*.SysMaint.*	
r	*.*.*	

rtdt

rw	Initializer.SysDaemon.*	a binary segment containing resource type descriptions.
r	*.Operator.*	
r	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.SysMaint.*	

sat

rw	Initializer.SysDaemon.*	the system administrator table (SAT), containing an entry for each registered project plus some per-system quantities; referenced at every login to validate the user's Project_id.
rw	*.SysDaemon.*	
rw	*.SysAdmin.*	
r	*.SysMaint.*	

sat.ht

r	*.SysDaemon.*	a hash table for the SAT.
r	*.SysAdmin.*	
r	*.SysMaint.*	

shift_config_change.ec

rw	Initializer.SysDaemon.*	an exec_com segment that is invoked at each shift or configuration change.
rw	*.SysDaemon.*	
rw	*.SysAdmin.*	
rw	*.SysMaint.*	

start_up.ec

rw	*.SysDaemon	The default start_up.ec file for users who do not
rw	*.SysAdmin	have a start_up.ec file.
rw	*.SysMaint	
r	*.*.*	

stat_seg

rw	Initializer.SysDaemon.*	system statistics, sampled at every accounting
rw	*.SysDaemon.*	update by the as_meter_ program and copied
rw	*.SysAdmin.*	daily by the copy_as_meters command (in the
r	*.SysMaint.*	crank).

system_start_up.ec

rw	Initializer.SysDaemon.*	an exec_com segment that is invoked
rw	*.SysDaemon.*	automatically when the answering service is
rw	*.SysAdmin.*	started.
rw	*.SysMaint.*	

ttt

r	*.*.*	the terminal-type definition table. Contains the
rw	*.SysDaemon	definitions of all valid terminal types recognized by the system.

vcons_tab

rw	Initializer.SysDaemon.*	the message coordinator's virtual console table.
rw	*.SysDaemon.*	
r	*.SysAdmin.*	
r	*.Operator.*	
r	*.SysMaint.*	

XXX.message

rw	Initializer.SysDaemon.*	the message coordinator input queue for source
rw.	*.*.z	XXX.

XXX.queue

rw	Initializer.SysDaemon.*	the message coordinator output queue for the
rw	*.SysAdmin.*	device whose channel name is XXX.
r	*.SysMaint.*	
r	*.SysAdmin.*	
r	*.Operator.*	

whotab

rw	*.SysDaemon.*	the public list of logged-in users.
r	*.*.*	

There are eight important directories in >sc1:

- admin_acs
- as_logs
- mc_acs
- pdt
- rcp
- syserr_log
- update
- volume_backup_accounts

Access to the admin_acs Directory

The >sc1>admin_acs directory contains ACS segments that control system administrative operations. Access to the directory should be set as follows:

```
sma         *.SysAdmin
sma         *.SysMaint
s           *
```

The IACL of the directory should be null to *.

The >sc1>admin_acs directory contains the following segments:

- send_daemon_command.acs
- tandd.acs
- send_admin_command.acs
- sat.install.acs
- rtdt.install.acs
- mgt.install.acs
- cdt.install.acs
- process_termination_monitor.acs
- bump_user.acs
- absentee_proxy.acs
- Fortran_hfp.acs
- set_proc_required.acs

The contents of each segment and the recommended access for each segment is listed below.

ACCESS TO THE send_daemon_command.acs SEGMENT

The send_daemon_command.acs controls the use of the send_daemon_command facility if the validate_daemon_command installation_parm is OFF. If the parm is on, this ACS is ignored. rw access grants users the right to use send_daemon_command. Users with send_daemon_command access may log daemons in and out, send them quits, and send them replies. If this segment is used, the ACL should be set as follows:

```
rw          *.SysAdmin
rw          *.SysMaint
null        *
```

ACCESS TO THE tandd.acs SEGMENT

The tandd.acs segment controls the ability of users to make Test&Diagnostic attachments of communications channels. The ACL of this segment should be:

```
rw          *.SysAdmin
rw          *.SysMaint
rw          *.HFED
null        *
```

ACCESS TO THE send_admin_command.acs SEGMENT

The send_admin_command.acs segment controls the use of the send_admin_command facility. Users with access to this facility can send any Multics command line or initializer command line to the Initializer process for execution. The ACL on this segment should be set as follows:

```
rw          *.SysAdmin
rw          *.SysMaint
null        *
```

ACCESS TO TABLE INSTALLATION ACS SEGMENTS

The following segments control the ability of users to install copies of the respective tables:

- sat.install.acs
- rtdt.install.acs
- mgt.install.acs
- cdt.install.acs

The access on sat.install.acs should be set as follows:

```
rw          *.SysAdmin
null        *
```

The acl on the remaining acs segments should be set as follows:

```
rw          *.SysAdmin
rw          *.SysMaint
null        *
```

ACCESS TO THE process_termination_monitor.acs SEGMENT

The process_termination_monitor.acs segment controls the ability of users to request a notification from the system each time that any process terminates. This is used by privileged processes to clean up their tables of entries relevant to a destroyed process. The access to this segment should be set as follows:

```
rw          Data_Management.Daemon.z
null        *
```

ACCESS TO THE bump_user.acs SEGMENT

The bump_user.acs segment controls the ability of users to request that other users be bumped (forcibly logged out) from the system. This facility is used only by the Data_Management daemon -- administrators and maintainers use send_admin_command. The access to this segment should be set as follows:

```
rw          Data_Management.Daemon.z
null        *
```

ACCESS TO THE absentee_proxy.acs SEGMENT

The segment absentee_proxy.acs in the directory >system_control_1>admin_acs controls the use of the proxy absentee facility. Any user with e access to this segment can submit proxy absentee jobs, that is, jobs to be run on behalf of, and under the User_id of, other registered users. Normally only daemons that operate the GCOS batch facility or the remote job entry (RJE) facility should be given e access to this segment.

The absentee_proxy.acs segment controls the ability of users to submit absentee jobs on behalf of other users. The access to this segment should be set as follows:

```
rw          RJE.SysDaemon.z (if you are using remote job entry)
rw          .SysAdmin
null        *
```

ACCESS TO THE Fortran_hfp.acs SEGMENT

The Fortran_hfp.acs segment controls the ability of users to make use of hexadecimal floating point. The access to this segment should be set according to site policy.

ACCESS TO THE set_proc_required.acs SEGMENT

The set_proc_required.acs controls which users may select alternate processors entry on which to run a process (using the set_proc_required command which uses hcs_$set_procs_required). rw access grants users the right to use the entry. The access to this segment should be set according to site policy.

Access to the as_logs Directory

The >sc1>as_logs directory contains the answering_service log, the admin log, and all logs created by the message coordinator. Set the access control list for this directory as follows:

```
sma         Initializer.SysDaemon
sma         .SysAdmin
sma         .SysMaint
s           ..*
```

Set the initial access control list to the as_logs directory as follows with the set_iacl command:

```
rw          Initializer.SysDaemon
r           .SysAdmin
r           .SysMaint
```

Access to the pdt Directory

The >sc1>pdt directory contains a project definition table (PDT) for each registered project. Whenever a user logs in, his entry is located in his project's PDT, and the user's attributes are used to initialize his process. Access to the PDT's under this directory is set automatically when the project attributes are changed.

Access to the pdt directory should be set as follows:

```
sma         *.SysDaemon
sma         *.SysAdmin
s           *.SysMaint
```

Access to the rcp Directory

The >sc1>rcp directory contains:

- All of the acs segments for communications channels.
- All of the acs segments for registered dial servers.
- If RCPRM IS NOT in use, all of the acs segments for RCP-controlled devices.
- If RCPRM IS in use, it will normally contain the acs segments for site-owned resources such as devices and dump volumes. It will also contain the RCPRM journal files.

The acl of the >sc1>rcp directory should be set as follows:

```
sma         *.SysAdmin
sma         *.SysMaint
sma         *.SysDaemon
s           *
```

Access to the syserr_log Directory

The >sc1>syserr_log directory contains the portion of the syserr log that has been copied out of ring zero but has not yet been copied by the crank to the history directory. Access to this directory should be set to correspond to the access to the syserr log segments in >sl1. Users with access to the syserr log can make extensive observations of the activities of the system and other users, including information about directories to which they have no access.

The directory ACL for >sc1>syserr_log should give s access to users who are permitted to read the log, sma access to the Initializer, and sma to *.SysAdmin.

The IACL of the >sc1>syserr_log directory should be r to users who are permitted to read the log, and rw to *.SysAdmin.

Access to the update Directory

Users place new copies of pdt's, sat's, rtdt's, cdt's, and mgt's in the >sc1>update directory for installation. The system automatically gives project administrators sufficient access.

The access to the >sc1>update directory should be set as follows:

```
sma        *.SysAdmin
sma        *.SysDaemon
sma        *.SysMaint
```

Append access should be set for any other users granted access to install one of the tables listed above, such as a project administrator.

Access to the volume_backup_accounts Directory

The >sc1>volume_backup_accounts directory is optional. If it exists, and if the volume dumper processes have sma access to it, then they will put segments into it that record all segments that they dump. The site may then write programs or exec_coms to use this information to charge users for volume dumping.

Access to volume_backup_accounts should be set as follows:

```
sma        *.SysDaemon
sma        *.SysAdmin
sma        *.SysMaint
sma        *.Daemon
```

Access on the mc_acs Directory

The >sc1>mc_acs directory contains the message coordinator acs segments used to control access to manipulate daemon processes. When the validate_daemon_command installation parm is enabled, mcacs segments in this directory control access to daemon processes logged in over specified mc source_ids. The recommended access to this directory is:

```
sma        *.SysAdmin
sma        *.SysMaint
```

CHANGING THE RING BRACKETS OF SPECIAL FILES

The default ring brackets of the following files are set to 4,4,4:

- installation_parms
- rtdt
- start_up.ec
- ttt
- whotab

Since all users must be able to access these files, set the default ring brackets on each of these files to 4,5,5 with the set_ring_brackets command (see the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64).

ACCESS TO THE admin DIRECTORY

All data segments pertaining to system administration are kept in the directory:

```
>udd>SysAdmin>admin
```

This directory is the working directory for the accounting administrators (who may have no access to the regular programming tools of the system and no private segments except a mailbox).

Access to the admin directory should be set as follows:

```
sma         *.SysAdmin
sma         *.SysDaemon
s           *.SysMaint
```

All segments in >udd>SysAdmin>admin should have their access set as follows:

```
rw          *.SysAdmin
r           *.SysMaint
r           *.SysDaemon
null        *
```

Other data segments kept in this directory include backup copies of billing data from previous months and other temporary segments. It is in this directory that the monthly bills and statistical reports are created.

Processed log files (copies of >sc1>log) are placed in the directory:

```
>udd>SysAdmin>admin>history
```

where they remain until they are deleted by the system administrator. They should be kept for about a month. Access to the history directory should be set as follows:

```
sma         *.SysDaemon
sma         *.SysAdmin
s           *.SysMaint
s           *.HFED
```

The SAT and the PDTs in the >sc1 directory are copied daily (by the accounting job known as the "crank") into the directory:

```
>udd>SysAdmin>admin>safe_pdts
```

for both backup and accounting purposes. These copies are used daily, to produce month-to-date charge summaries, and, at the end of a billing period, to produce the bills and to reset the usage charges in PDTs stored in >sc1>pdt.

Access to safe_pdts should be set as follows:

```
sma         *.SysAdmin
sma         *.SysDaemon
s           *.SysMaint
```

Copies of the usage totals and the projfile and reqfile segments for previous months are kept in another directory:

```
>udd>SysAdmin>admin>HF
```

for some time after billing, so that the billing can be rerun if necessary. These segments should be dumped to tape and deleted after a reasonable period. Access to the HF directory should be set as follows:

```
sma       *.SysDaemon
sma       *.SysAdmin
s         *.SysMaint
```

Check pointed copies of the RCPRM resource registries copied by the crank are kept in:

```
>udd>sa>a>safe_registries
```

Access to the lib Directory

Site-dependent system administration tools are kept in the directory:

```
>udd>SysAdmin>lib
```

All segments in >udd>sa>lib should have the following access:

```
r         *.SysAdmin
r         *.SysDaemon
r         *.SysMaint
null      *
```

Note that the update_seg command should be used to install new segments in this directory.

All system administrators should use the add_search_rules command to add the >udd>SysAdmin>lib directory to their search rules as part of their start_up.ec.

THE DAEMON DIRECTORY

The >daemon_dir_dir directory contains the following directories:

- cards - a storage pool for card deck image segments read by the system card input process.
- gcos - used by the GCOS daemon
- io_daemon_dir - holds all I/O daemon data bases
- volume_retriever - holds volume retrieval queues
- volume_backup - holds volume backup information

Access to the cards Directory

The storage pool for card deck image segments consists of a subtree of the directory hierarchy, which is headed by >daemon_dir_dir>cards. One access class directory for each access class is contained in the cards directory. Storage is always allocated within the access class directory that corresponds to the caller's authorization. Person directories are contained in the appropriate access class directory. A person directory is created for each person who needs temporary storage. A person directory contains all segments and multisegment files for a given person at a given access class. For example, if a user with Person_id TSmith is at system_low, the following directory is allocated for his card deck image segments:

```
>daemon_dir_dir>cards>system_low>TSmith
```

Access to >ddd>cards should be set as follows:

```
sma       *.SysDaemon.*
sma       *.SysAdmin.*
sma       *.SysMaint.*
```

In addition, sma access should be given to the directory for any other IO daemon that performs card input, including HASP workstation simulators. Note that access to subdirectories of >ddd>cards is set automatically.

Access to the gcos Directory

Access to >ddd>gcos should be set as follows:

```
sma       GCOS.SysDaemon
sma       *.SysAdmin
sma       *.SysMaint
```

Access to the io_daemon_dir Directory

The >daemon_dir_dir>io_daemon_dir directory contains a set of administrative data bases and working storage used to direct the activities of the I/O coordinator process and the various device drivers.

The main data base, iod_tables, is set up by a system administrator. Most of the other segments and directories are created and maintained by the I/O coordinator acting on information contained in the iod_tables segment.

Access to >ddd>idd should be set as follows:

```
sma       *.SysDaemon
sma       *.SysAdmin
sma       *.SysMaint
s         *.*.*
```

There are several directories contained in io_daemon_dir. One, coord_dir, is used for working storage by the I/O coordinator. The other directories are site dependent; a directory is created and named for each active device.

Access to the volume_retriever Directory

Access to >ddd>volume_retriever should be set as follows:

```
sma       *.SysDaemon
sma       Volume_Retriever.Daemon
sma       *.SysAdmin
sma       *.SysMaint
s         *
```

Access to the volume_backup Directory

Access to >ddd>volume_backup should be set as follows:

```
sma       *.SysAdmin
sma       Volume_Dumper.Daemon
sma       Volume_Reloader.Daemon
sma       *.SysMaint
```

Access to the lv Directory

The >lv directory contains supervisor data bases used for registration and access control of logical volumes and master directories. Two supervisor subsystems, logical volume management and master directory control, use these data bases to manage the logical volume environment. These subsystems are implemented in ring 1.

Logical volume management maintains a list of all logical volumes registered on the system, and their attributes. Logical volume management knows, for example, which physical volumes belong to a logical volume. When a request to mount a logical volume is received, the subsystem cooperates with ring 0 volume management to ensure that all required physical volumes are mounted.

Master directory control manages directories whose segments reside on logical volumes other than the root logical volume (RLV). Such directories are called master directories. A master directory is simply the point where the hierarchy "branches out" to another logical volume.

The supervisor data bases reside under the directory >lv either as segments in the hierarchy, as described below.

Volume Registration Segments

There is one segment for each logical volume known to the system. Attributes maintained include which physical volumes constitute the logical volume. Each segment has the names:

lv.LVNAME
where LVNAME is the name of the logical volume

lvid.LVID
where LVID is a character representation of the unique identifier of the logical volume

pv.PVNAME
where PVNAME is the name of a physical volume belonging to the logical volume (there is one for each such physical volume)

pvid.PVID
where PVID is a character representation of the unique identifier of a physical volume belonging to the logical volume (there is one for each such physical volume)

The volume registration segments are used by ring 1 volume management in the initializer process. They are ring 1 segments with rw access to the initializer and general r access.

Access Control Segments

There is an empty access control segment (ACS) for each logical volume known to the system. The access control list (ACL) for each segment defines the permissions to use the respective logical volume. Each segment has the name LVNAME.acs, where LVNAME is the name of the logical volume.

To attach a private logical volume, a user must have rw access to the ACS for that logical volume (rw access to a public logical volume is meaningless since a user does not attach a public logical volume explicitly). To have executive access to a logical volume, a user must have e access to the ACS for that volume. Executive access is required to perform the following functions on a given logical volume:

set, change, or delete volume quota

set the owner of a logical volume

set quota on a master directory on a logical volume

list all master directories on a logical volume

Master Directory Control Segments

There is a segment for each logical volume known to the system. Each segment defines all master directories on the logical volume and all users who have volume accounts on the volume. Each segment has the name LVNAME.mdcs, where LVNAME is the name of the logical volume.

Master directory control segments are ring 1 segments with general rw access.

For security purposes, the following procedures should be observed in the administration of a master directories:

Only system administrators should have executive (e) access to a logical volume

Only system administrators can create master directories (i.e., only system administrators can have quota accounts).

Only system administrators can "own" a master directory.

Because of a covert channel that exists in master directory control, information can be entered into mdc at system high and retrieved at system low thereby violating the AIM boundary. For this reason, it is recommended that at sites concerned about AIM only system administrators should be allowed to change the contents of MDC segments.

Access to the site Directory

The >site directory is the root of the subtree of all site-specific system-wide information that is not necessary to boot the system and start the answering service. The access to the >site directory should be set as follows:

```
sma         *.SysAdmin
sma         *.SysMaint
s           *
```

You may create directories as needed under >site for site-specific libraries or facilities. The following subdirectories of >site are reserved for use by Honeywell-supplied software:

- subsystem_usage_dir (ssudir)
- mail_system_dir (mail_system mlsys)
- forum_dir (fd forum)
- Data_Management (data_management dm)

Access to the subsystem_usage_dir Directory

If the >site>subsystem_usage_dir directory exists at your site, then ssu_ subsystems that call ssu_$record_usage will record their usage here.

Access to the >site>subsystem_usage_dir directory should give a access to all users whose usage is to be recorded. The SysMaint project (or others who are allowed to examine and delete usage records) should be given sma.

Access to the mail_system_dir Directory

The >site>mail_system_dir directory contains the system mail table, which records the mailing addresses of all registered users.

Access to the >site>mail_system_usage_dir directory should be set as follows:

```
sma         *.SysAdmin
sma         *.SysMaint
s           *
```

Access on the mail table which resides in this directory is set automatically by the mail table software to rw for *.*.*.

Access to the forum_dir Directory

The >site>forum_dir directory is used by the forum interactive meeting subsystem. By default, it is the site public meeting directory. Forum also stores some internal data here. Users who are permitted to create official site-wide meetings should be given sma access to this directory. All others should have s.

Access to the Data_Management Directory

The >site>Data_Management directory contains per-AIM data management system directories. Access to >site>Data_Management should be set as follows:

```
sma       Initializer.SysDaemon.*
sma       Data_Management.Daemon.*
sma       *.SysDaemon.*
sma       *.SysAdmin.*
sma       *.SysMaint.*
```

ACCESS ON GATES

The system uses the ring mechanism (see the Programmer's Reference Manual, Order No. AG91) to protect internal data bases. Users call gates to request services from the system. The system gates are organized into several gate segments. You must set access to these gate segments appropriately to control access to privileged services.

There are two types of gates:

- hardcore gates
- online gates

Hardcore gates are located in >system_library_1. Access to hardcore gates should be set by putting entries in the system_start_up.ec to set their ACL. Online gates are located in system_library_standard, system_library_tools, or system_library_unbundled; their ACLs are maintained in the usual way with the 1_sat_acl command. For a complete list of all gates and the proper access that should be set for each gate, see Section 23.

You control access for many privileged or semiprivileged gate segments that allow the user to request some specific supervisor action. Each of these gates has an access control list that defines the set of users who can call it. For example, the mhcs_ gate (metering_hcs_) permits you to ask the question, "How many page faults have been taken on a segment?" If you can ask this question, you can transfer information from system_high to system_low thereby violating AIM boundaries. However, access to this gate permits a user to tune an application by knowing how many page faults have been taken on a segment. For this reason, only temporary access should be given to the mhcs_ gate for the user if the site is not overly concerned about AIM covert channels.

Project administrators can arrange for all users on their project to log in into a higher ring than usual, and then provide gates from this higher ring to selected services in the standard user ring. Thus, project administrators can exercise complete control over which parts of the Multics environment a user process can access.

If a project administrator wants to maintain a data base or subsystem and have the ring mechanism protect its access and integrity, he/she may create the subsystem in ring-4 and change the PDT entry for members of his/her project so that they log in at, for example, ring 5. Then the administrator must write a gate program that allows access to the ring-4 subsystem from ring-5 in a controlled way. The users logging in at ring-5 can thus access the ring-4 data base set up by the project administrator.

ACCESS ON PROJECT DIRECTORIES

Access to the project directories is set automatically by the software that creates and updates project attributes. Additional access can be set on the project directory as necessary. At an open site, s access to *.*.* is appropriate. Otherwise no access should be set to *.*.*.

ACCESS ON NONADMINISTRATIVE SYSTEM DATA BASES

The principal Multics data bases are:

- >sl1>sys_info
- >sl1>time_table
- >sl1>config_deck
- >sl1>io_config_data

Access to sys_info and time_table are managed automatically.

The config_deck and io_config_data show what the hardware configuration of the system is, including how much hardware is actually running. The access defaults to r to *. To change the default, you must put hp_set_acl commands in the ssu.ec.

Access to sys_info and time_table is set in the standard system hardcore tape and you do not have to specifically set access to these data bases.

ACCESS ON I/O DAEMON QUEUES

The I/O daemon queues are where I/O daemon requests are placed by users before being processed by the I/O daemons. Because the I/O daemon queues are message segments, access to the queues is determined by extended access modes (adros) using the set_acl command.

Give the IO.SysDaemon identity full extended access, i.e., add, delete, read, own, and status (adros) to all queues, by typing:

```
set_acl **.ms adros IO.SysDaemon.*
```

or

```
set_acl QUEUE_NAME.ms adros IO.SysDaemon.*
```

For standard system queues (i.e., queues for which the driver_userid of the corresponding request type is IO.SysDaemon), give aros permission to all users by typing:

```
set_acl **.ms aos *.*.*
```

or

```
set_acl QUEUE_NAME.ms aos *.*.*
```

Otherwise, the assumption is made that the queues are dedicated to the particular project named in the driver_userid. In this case, give aros permission just to users of that project. The list_acl command can be used to list the extended access on the queues and the set_acl command can be used to change the extended access on the queues.

UNDERSTANDING THE ACCESS ISOLATION MECHANISM

The purpose of access isolation mechanism (AIM) controls is to provide a high degree of security to sites that have a need to separate sensitive information into different levels and categories of sensitivity. For detailed information on the access isolation mechanism, see the *Multics Programmer's Reference Manual*, Order No. AG91. Also see Section 43, "Data Management Administration", for specific considerations in setting up Data Management for sites running AIM.

Levels and Categories

There are several fields in >sc1>installation_parms used by AIM. These fields are long and short names for each of eight sensitivity levels and 18 access categories used by the AIM software. In addition, another field, called the access_ceiling, specifies a limit on the levels and categories that may actually be used.

Three keywords used in the ed_installation_parms command (level_names, category_names, and access_ceiling) enable a system administrator, or a system security administrator, to print, change, or retype the specified fields. In the case of level_names and category_names, the retype operation requests new names for all eight levels and 18 categories (both short and long names). The change operation first requests a level number (from 0 to 7) or a category number (from 1 to 18), and then requests new long and short names for that level, or category, only. In the case of the access_ceiling, the change and retype operations are equivalent; the editor requests a level number (from 0 to 7) and a category set (specified by six octal digits -- each digit representing three category bits). If the _ith bit is on, then category _i is included in the category set of the access ceiling.

For the system to function properly, all levels less than or equal to the level of the access_ceiling, and all categories contained in the category set of the access_ceiling, must have both a long and a short name. The only exception is level 0; it may be left unnamed (reply "." to the editor when it asks for a name for level 0). If level 0 is left unnamed, processes whose authorization is "level 0, no categories" (the system_low authorization), see no difference in the operation of the system from operation without AIM controls.

No one can use AIM unless certain commands are used by the system administrator (or the system security administrator) to set access class and authorization attributes for users and projects in various system tables (PNT, SAT, PDT.). Listed below are the commands that can be used with the various system tables.

```
System Table        Command

PNT                 new_user

SAT                 new_proj
                    edit_proj

PDT                 an editor
                    cv_pmf
                    install

CDT                 an editor
                    cv_pmf
                    install
```

Note that when using the install command, the "-a" control argument must be used for AIM attributes to take effect.

Changing the Names of Levels or Categories

If AIM is enabled at your site, users will create "marked" objects. That is, objects marked with specific levels and categories. If you change any of these names, you can potentially confuse the user population because the names they had been used to using will not work any longer. In other words, once you assign names and meanings to levels and categories, people will create objects based on that criteria and those objects will persist. Changing the names will not change the objects that have been created. Therefore, it is not recommended that you change the name or meaning of any level or category.

If you assign an AIM category or level, it should not be changed. If another category or level is needed, create a new one.

Turning on AIM

The Access Isolation Mechanism is always present in a Multics system. All the access control checks that are AIM related are always made. However, until you have differentiated the system into the various AIM access categories and classes everything in the system runs in the same category and class. The process of activating AIM involves establishing more than one category for information and granting individuals access to the various categories.

To activate AIM, the following steps or procedures should be followed:

1. Decide the names of the various levels and categories, and establish them in the installation_parameters segment.

2. Assign administrative projects and persons the ability to log in at "system_high" by changing the project registration attributes in the SAT, PDT, and the PNTs for all of the administrators and their projects (see "Project Registration").

3. Use the reclassify_sys_seg command to change all I/O daemon queues, absentee queues, volume backup queues, and personal mailboxes from system_low to system_high.

4. If RCPRM has not been configured for devices, configure it so that the system devices such as tapes and disks have the full range from system low to system high.

VOLUME TYPE	ACCESS
Backup tape volumes for hierarchy dumper	system_high
Volume dumper tape volumes	system_high
User Tape volumes	system_low

5. Move system overhead processes to system_high. The system overhead processes that must be at system_high are:

   - Hierarchy backup dumper
   - I/O daemon
   - Volume dumper

   To change the hierarchy backup dumper to system_high, change the PMF to allow a login authorization of system_high and change the admin.ec to log it in at system_high with -auth system_high. Also, change the PNT of the Dumper.Daemon, Backup.Daemon, and IO.SysDaemon to have a login range of system_low to system_high. To change the I/O daemons, you must modify the I/O daemon tables to state the AIM categories of the printer devices and other devices specified in the tables.

6. At this point in the AIM creation process, determine what part of the system hierarchy you want to reclassify at other than system_low. It is impractical to reclassify large amounts of hierarchy. If you want to upgrade an existing piece of your hierarchy, it should be dumped on tape and brought back with the upgraded classification. It is recommended that project directories be set to the lowest authorization of its lowest user; otherwise, some users will not be able access the directory. A user's home directory should be set at the lowest authorization at which a user is allowed to log in. You can only create things at your lowest authorization. If a user's home directory is not at the user's lowest authorization, then the user will not be able to create an abbreviation file or perform other system functions. Individuals who want to maintain classified information should create upgraded hierarchies with the "create_dir -access_class" command.

   If a project requires an upgraded classification, the contents of the project directory should be dumped on tape, deleted from the system, recreated with the create_dir command and the contents reloaded from the tape.

7. To set authorizations for users of undelegated projects:

   - Assign the project a new authorization with the edit_proj command and install it with the install command.
   - Edit the authorization statements in the PMF
   - Edit the PNTs of individual users to assign authorization levels.

   To set authorizations for users of delegated projects:

- Change the SAT with edit_proj command
- Project administrator is responsible for changing the authorizations of projects and users for whom he is responsible.

The access_class_ceiling parameter allows an installation to restrict the number of categories, and the maximum level used to less than the limit of 18 and 8, respectively. To change the access_class_ceiling, use the ed_installation_parms command and set the access__class_ceiling parameter. Once set, shut the system down and reboot. However, any PNT, PDT, SAT, and CDT entries which were set up with "system_high" as one of the AIM levels will now have to be changed to define the "new" system_high. System_high is defined as being the access_class_ceiling. If you redefine access_class_ceiling, then all those AIM levels you have set at system_high (before the redefinition) are now no longer system_high.

Since these steps require a fair amount of work, the access_class_ceiling parameter may be set higher than necessary without much loss of control over AIM. Therefore, it is recommended that the access_ceiling be set to the highest possible value:

```
level=7, categories=777777
```

The keyword system_high (when passed to the convert_authorization_ subroutine described in the *Multics Subroutines and I/O Modules* manual, Order No. AG93) returns the value of the access_ceiling. See the description of the ed_installation_parms command for instructions on setting access_ceiling.

AIM Level of Library Directories

The AIM access class range for system library directories should be set at system_low. Library directories that contain sensitive data, should be set appropriately.

AIM Level of SysAdmin Directories

The AIM access class range for all directories in >udd>sa should be set at system_low.

AIM Level of I/O Daemon Queues

To enable the AIM features of the I/O daemon at a site that has been operating without using AIM, follow the instructions below.

1. Read the I/O daemon sections in this manual.
2. Raise the access classes of the I/O daemon queues to system_high. This is done by issuing the following commands:

```
cwd >ddd>idd
reclassify_sys_seg ([segs **.ms]) system_high
```

You must have access to system_privilege_ to use the reclassify_sys_seg command.

3. Modify iod_tables.iodt to define device classes if desired. If no device classes are wanted, be sure to set the max_access_class for each request type to system_high.

4. Change system_start_up.ec so that the coordinator is logged in at system_high authorization. This is done by the following line in system_start_up.ec:

```
sc_command login !0 SysDaemon cord -auth system_high
```

   Similarly, make sure that all drivers are logged in at the correct authorization. If no device classes are defined, then the correct authorization for a driver is the max_access_class of the default request type for the driver (which should normally be system_high). If device classes are defined, then the correct authorization for a driver is the max_access_class for the default device class for the driver.

5. Compile the new iod_tables.iodt. The next time the coordinator is logged in, it begins using the new iod_tables. At this point, AIM features of the I/O daemon are enabled. Drivers thereafter must have appropriate authorizations.

AIM Level of Absentee Queues

To allow processes of authorizations greater than system_low to submit absentee requests, raise the access class of the absentee queues to system_high. To do this, type the following commands:

```
cwd >sc1
reclassify_sys_seg ([segs absentee_*.ms]) system_high
```

You must have access to system_privilege_ to use the reclassify_sys_seg command. You must log in at system_low with a maximum access class of system_high.

Access on Backup Processes

To dump directories that have access classes greater than system_low, the hierarchy backup processes, Backup.SysDaemon and Dumper.SysDaemon, must be run at system_high authorization. This requires that the following steps be taken.

1. Create the following two upgraded directories with system_high access classes:

```
>udd>SysDaemon>Backup>system_high
>udd>SysDaemon>Dumper>system_high
```

   These are used as working directories where the backup processes can create dump maps. Set quota on these directories to 1000.

2. In each system_high directory, create links to all dump control files contained in the respective containing directories.

3. Create a start_up.ec for Backup.SysDaemon and Dumper.SysDaemon that automatically changes the working directory to the system_high directory when either process logs in at system_high authorization. The following start_up.ec can be used:

```
&command_line off
&if [equal [user auth] [user max_auth]]
&then &goto system_high
ioa_ "Warning: ^a not at system_high" [user name]
&quit
&label system_high
cwd system_high
&quit
```

Place this start_up.ec in the >udd>SysDaemon>Backup and >udd>SysDaemon>Dumper directories.

4. Change system_start_up.ec so that Backup.SysDaemon is logged in at system_high authorization. This is done by the following line in system_start_up.ec:

```
sc_command login Backup SysDaemon bk
   -auth system_high
```

In the case of volume backup, for the volume retriever to handle retrieval requests that have access classes greater than system_low, the volume dumper and volume reloader processes must also be run at system_high authorization. This is accomplished by changing system_start_up.ec so that Volume_Dumper.Daemon is logged in at system_high authorization. The new line in system_start_up.ec is:

```
sc_command login Volume_Dumper Daemon vd -auth system_high
```

CORRECTING SECURITY SERVICE PROBLEMS

Security out_of_service problems arise when AIM restrictions are being violated because of inconsistencies in the file system. To correct "security out-of-service" problems, use one or more of the following commands:

- priv_move_quota - lets you get quota in or out of an upgraded directory that is not empty. Normally, you would create an upgraded directory empty with "cd <dir_name> -auth -quota" and the directory gets a specified amount of quota and that is the amount of quota the directory will always have. If you have access to system_privilege_, you can use priv_move_quota to give the upgraded directory more quota.

- reclassify_sys_seg - takes a ring_1 multiclass segment which may be a mailbox, message segment, or a volume registration segment and changes the maximum authorization for information in it. It is used when you have a daemon queue, absentee queue, or some other multiclass segment that was created with a maximum authorization that is not system_high and which you want to be system_high.

- reclassify_dir and reclassify_seg - are used to repair file-system damage. When you get a directory or segment whose access marking is inconsistent, you can upgrade the directory or segment with the reclassify_dir command and reclassify_seg command, respectively.

- reset_soos – is used as part of the reclassification of a directory or segment. When the Salvager detects that something has become inconsistent (a segment is not consistent with its directory, a directory is not consistent with its parent directory, or an upgraded directory does not have terminal quota), the Salvager sets a "security out-of-service switch" for the segment or directory. Once this switch is set, no one can access the segment or directory until you turn off the security out-of-service switch (with reset_soos) or you have the "soos" privilege. Therefore, the reset_soos command is used after you have reclassified an inconsistent directory or segment to reset the security out-of-service switch.

- set_system_priv – gives you the right to and correct arbitrary difficulties. For example, if you have a completely damaged hierarchy and your job is to get rid ot it; or, you have to examine something at a different authorization than your current authorization. set_system_priv lets you turn on arbitrary privileges that allow you to examine and modify things without any regard to AIM boundaries. This is very dangerous and you must be careful not to produce any security violations.

SPECIFYING AUTHORIZATIONS FOR USERS

The default authorization values can be changed with the new_user command, as follows:

```
new_user$cga Person auth NEW_AUTH_RANGE
```

The default values for AIM-related attributes for new users are set in the PNT at the time the user was registered as follows:

```
person authorization range:   system_low:system_high
default person authorization:   system_low
audit flags:   none
```

SPECIFYING AUTHORIZATIONS FOR PROJECTS

Only a system administrator or a system security administrator may alter the default values of the nondiscretionary access control attributes for the new project with the edit_proj command. The default values for AIM-related attributes for new projects are set in the SAT as follows:

```
authorization:   system_low:system_low
audit:   none;
```

SECTION 19

ASSURING THE SECURITY OF RCPRM

System I/O resources can be managed with the Resource Control Package (RCP). RCP lets you enable the Resource Management facility (RCPRM) which is an optional part of the Resource Control Package (see Section 8). Enabling Resource Management gives you greater flexibility in system resource management, especially the management of tape volumes.

ACCESS TO ACCESS CONTROL SEGMENTS

RCPRM uses access control segments to control user access to each resource that it controls. The name of an ACS consists of a name plus the suffix (e.g., prta.acs). The RCPRM access control segments for each device are usually located in >system_control_1>rcp but can be located anywhere you choose.

Access to Devices Managed by RCPRM

To create an ACS for a device, you must:

1. Change your working directory to the rcp directory:

```
cwd >sc1>rcp
```

2. Create an acs segment for the device:

```
cr device_name.acs
```

3. Set the maximum length of the acs segment to 0, as follows

```
sml <device_name.acs> 0
```

4. Set access to the acs with the set_resource command, for example:

```
set_resource tape_drive tape_01
   -acs_path >sc1>rcp>tape_01.acs -priv
```

Access to Volumes Managed by RCPRM

For disk or tape volumes that are to be owned by the system and used for system purposes such as incremental backup, you should do the following to assure their security:

1. Acquire them for the system by registering with "-owner system"

2. Turn on their allocation flag by typing:

```
set_resource <volume_name> -alloc on -priv
```

3. Change your directory to the rcp directory that contains the acs segments for resources by typing:

```
cwd >scl>rcp
```

4. Create the acs segment for the volume:

```
cr <volume_name>.acs
```

5. Set the access on the acs for the volume:

```
sa <volume_name>.acs rw *.SysDaemon.*
sa <volume_name>.acs rw Volume_Dumper.Daemon.*
sa <volume_name>.acs r *.Operator.*
sa <volume_name>.acs rw *.SysMaint.*
sa <volume_name>.acs rw *.SysAdmin.*
sa <volume_name>.acs null *.*.*
```

AIM FOR RCPRM

Access class ranges are used by RCP to specify that a process within a range of authorizations can use a particular resource.

An access class range is simply a pair of AIM access classes separated by a colon. The first value of the pair is the minimum access class and the second is the maximum access class. If only a single access class is specified when an access class range is expected, the minimum and maximum access class values are both the same (i.e., a range of one value). The second access class of the pair (the maximum) must be greater than or equal to the first (the minimum) according to the aim_check_ subroutine (see the *Multics Subroutines and I/O Modules* manual, Order No. AG93).

A process with authorization of:

```
level2,category1
```

would not be able to use a resource whose access class range was:

```
level1,category1,category2:level3,category1,category2,category3
```

where level3 is greater than level2, which is greater than level1. This is due to the fact that the authorization of the process is isolated from the minimum of the access class range. To allow this process access to the resource in question, the range would have to exclude category2 or the user would have to have category2 authorization.

In general, to include categories within an access class range, both the minimum and maximum must include the categories desired. If combinations of categories are desired, the minimum should list only required categories and the maximum should include all categories allowed. For example, the access class range:

```
level1,category1:level3,category1,category2,category3
```

allows read and write access to any level1, level2, or level3 process with category1 and any combination of category2 and category3.

RCP Effective Access

Viewed separately, each type of access control answers the same question, "What access does a particular process have for a particular item?" The access mode granted a process to a resource by discretionary access control (the ACL) is known as the raw access mode.

The way RCP determines effective access to a resource for a process differs from the regular Multics method of determining effective access as follows:

1. The effective access to the ACS for the resource is determined as for any segment. If the ACS does not exist, the user appears to have read, execute, and write access if he is the owner of the resource, or null access if he is not the owner.

2. The current authorization of the process is compared to the maximum access class of the resource. If write access is not allowed (as defined by the write_allowed_ subroutine) then write and execute access are denied and only read is allowed.

3. The current authorization of the process is compared to the minimum access class of the resource. If read access is not allowed (as defined by the read_allowed_ subroutine) then all access is denied. The resulting access is termed the RCP effective access to the resource.

4. To use a device, the RCP effective access must include both read and write to that device (a restriction not imposed on volumes).

For example, the following table illustrates some examples of RCP effective access. In the examples below, L1, L2, L3 and L4 represent sensitivity levels and c1, c2, c3, and c4 represent categories. *This discussion mostly concerns devices; volumes should never be given a multiclassed access class range because multiclass volumes are not allowed.*

Table 19-1. RCP Effective Access

Effective Access to ACS	Current Process Authorization	Resource Access Class Range	RCP Effective Access
rew	L1	L1:L3	rew
re	L1	L1:L3	re
rew	L1	L2:L3	null
rew	L3	L2:L3	rew
rw	L4	L2:L3	r
re	L4	L2:L3	r
rw	L2,c1	L1:L4	r
rw	L2,c2	L1,c1:L4,c1,c2	null
rw	L2,c1,c3	L1,c1:L4,c1,c2	r
rw	L2,c1	L1,c1:L4,c1,c2	rw

A user must have write RCP effective access to the resource named to perform any modification on the status of the resource. In addition, the user must have execute effective access to the resource named to modify protected attributes. Only the accounting owner may modify the ACS path.

For more information on AIM, access classes, authorizations, and comparisons involving access classes and authorizations, see the *Multics Programmer's Reference Manual*, Order No. AG91. The access class range mentioned above is specified by the -access_class control argument, which can be specified in the register_resource command, and the acquire_resource and set_resource commands (described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Manipulating RCP Effective Access

Since the access control mechanisms described above operate together to determine the RCP effective access of a process, there are actions that the user, as well as an administrator, can perform to control this effective access:

1. Create an ACS via the create command.

2. Set the ACL for that segment using the set_acl command to add desired ACL entries, and the delete_acl command to delete entries. (The above three commands are described in the *Multics Commands and Active Functions* manual, Order No. AG92.)

3. To further affect the ACS, modify its ring brackets by using the set_ring_brackets command (described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64). The system security administrator sets the AIM access class range of the resource itself at the time it is registered using the register_resource command, and can change it by using the set_resource command.

Note that the ACS segment must be in a system_low directory because the AIM ranges for the volume are determined by the registry not from the ACS. If you do not put them in a system_low directory, the a user trying to read the ACS will not be able to read it.

SECTION 20

COMMUNICATION CHANNEL SECURITY

Communication channels are defined by you in the Channel Master File (CMF.cmf) located in >udd>sa>admin. Communication channel security is controlled by access control segments that you must create in the directory >scl>rcp. The rcp directory is used to contain the ACSs for communication channels.

Depending on the value you specify in the check_acs statement in the Channel Master File for the channel, any or all of the following can be checked for a channel ACS:

- dial out requests for autocall channels
- a privileged attach of a slave channel
- login

If the check_acs statement is set in the CMF for the channel, the ACS is checked when a user logs in to determine if the user has access to the channel. The user can log in only if access is set for the specific user for that channel. This feature permits restricted access to specific channels. If the check_acs attribute is not set, dial out requests and privileged attaches of slave channels are checked against the ACS but log ins are not checked and users can log in over any available channel. Also, if check_acs is not set, user names are not associated with the audit.

By default, the system allows anyone to log in over all login channels. It allows any channel to be dialed to any process via the dial mechanism. The system requires the process to have rw access to the ACS segment associated with the communications channels to priv_attach or dial_out to the channel. The channel ACS is:

```
>scl>rcp>CHANNEL_NAME.acs.
```

The system always requires a user to have the dialok attribute to priv_attach a channel, dial_out a channel, or become a dial server.

You can make the system more or less restrictive by using the check_acs statement in the CMF. You can require users to have access to the channel acs segment to log in over the channel or to attach the channel via the dial server mechanism.

If you are using AIM, you must specify the AIM characteristics of communications channels using the access_class statement of the CMF.

ACCESS FOR DIAL-OUT CHANNELS

The ACSs for dial out channels are in the directory >system_control_1>rcp.

The segment names have the form:

```
<channel_name>.acs
```

To set access to one of the dial-out channels shown in the example above for all users, type:

1. cwd >sc1>rcp

2. create b.h000.d01.acs

3. sa b.h000.d01.acs rw *.*.*

The procedures above assign access to one of the dial-out channels defined in the CMF. This procedure can be repeated for each channel.

To originate an outward telephone call from a particular tty channel, a user must have rw access to the ACS. The user must also have the dialok attribute in his PDT (see "Project Registration").

ACCESS FOR REGISTERED DIAL IDENTIFIERS

A dial identifier is a name that a person connecting to the system can specify with a dial-preaccess command to connect their terminal to some process instead of logging in a new process. It allows their terminal to become an attached device of another process. Dial identifiers can be either unregistered or registered.

Any user can use an unregistered dial server. No special access is required to use an unregistered dial server except the dialok attribute in the PNT and access to the channel to be used. For a user to connect to another process using an unregistered dial server, the command line:

```
dial <dial_id> <user_id>
```

where dial_id is the dial_id of the process with which you want to connect and user_id is the user_id of the user whose process to which you want to connect.

For system applications, there is a registered dial server which is unique to the entire system. Therefore, only the dial_id is required.

A registered dial identifier would be used in the following instance: if you had a program on the system that was part of the mail system that allowed other systems to dial up on the phone and exchange mail, you would not want to create a separate process for this purpose because it would require a name and password, etc. The registered dial identifier would allow the other system to issue, for example, the command "dial smtp" where smtp is a registered dial identifier belonging to the process called Network_Server.Daemon. This causes Network_Server.Daemon to receive a wakeup requesting a channel attachment. It attaches the channel and performs I/O to it in the same manner it would for any I/O device.

Whenever you have a process that requires a registered dial identifier, usually an I/O daemon, you must create an ACS in the directory >system_control_1>rcp in the same manner as described above for Dial-out channels.

The segment names have the form:

```
dial.<dial_id>.acs
```

To use a particular registered dial identifier, assign a user rw access to the ACS.

ACLS FOR COMMUNICATIONS CHANNELS

The ACSs for communications channels must also be created in the directory >system_control_1>rcp in the same manner as described above for Dial-out channels.

The segment names have the form:

```
<channel_name>.acs
```

The "check_acs" flags in the Channel Master File (CMF) control who has access to the communications channels, as described below:

login	The user must have rw access to the acs segment to log in over this channel.
slave	A user entering the "dial" or "slave" preaccess command must specify -user NAME (and optionally -access_class) to specify their user id, and must have rw access to the acs segment.
priv_attach	The user must have rw access to the acs segment to perform a privileged attach on this channel.
dial_in	A user accepting dials via dial_manager_$accept_dials or dial_manager_$registered_server must have rw access on the acs segment for the channel to be connected to her process.
dial_out	The user must have rw on the acs segment for this channel to perform a dial_out.
all	All of the above.

AIM FOR COMMUNICATION CHANNELS

You can assign an authorization range to a specified communications channel by using the access_class statement in the channel master file to specify the authorization range.

The authorization can be specified as a single value, in which case the channel is usable only by users with the specified authorization. The authorization can be specified by a minimum and maximum value, in which case the channel is usable only by users with an authorization equal to or greater than the minimum value and equal to or less than the maximum value.

If the access class statement is not specified, the value is assumed to be that specified (or defaulted to) by the Access_class global statement.

You must be aware that the system cannot establish the authorization of a user for any channel except channels identified as multiplexer_type sty. For this reason, it is recommended that all channels except those identified as multiplexer_type sty should be specified with a single access_class value. The only exception to this recommendation is for login service type channels. These channels should be assigned an authorization range sufficient to cover the users who are to be permitted to log in over the channel.

User Identification and Authentication

By site option, the slave preaccess command and dial preaccess command can be made to require the use of the -user control argument by using the check_acs statement in the CMF, or the global Check_acs statement if you want to require it for all channels. The Answering Service prompts for the password of the user name given. The -access_class control argument is optional. The user's PNT and PDT access class restrictions apply to the access class given with -access_class. For example:

```
dial internet -user Margulies.Multics -auth system_high
```

Access Class Extensions

For login service class channels, set the access_class to be the range of authorization at which users may log in over this channel. This will be determined by the physical security characteristics of the communications line and the terminal. For example, terminals in a top_secret terminal room might be access_class: "system_low:top_secret"; or access_class: top_secret; Terminals in an unsecured terminal room, or dial up lines, would be access_class: system_low .

For slave and autocall channels other than sty's, set the access class to be the single level that describes the information storage device at the other end.

For sty channels, set the access class to be the range of processes that may use this channel. If you define access class ranges for sty's, there may be resource contention between more and less authorized processes. If you need to avoid this, create many different sty subchannels and make each single class. The exact sty subchannels that you define will depend on the particular application at your site.

The Communications Privilege

To support the construction of trusted applications, a communications (comm) privilege is available. A process with the comm privilege can attach a communications channel at an access class less than or equal to its authorization. Thus, a process dialing out with the comm privilege can request that the access class of a channel be set to any single level less than or equal to the process's current authorization. A process making a privileged attachment or accepting dials with the comm privilege can attach any channel whose access class (as determined by a preaccess command), protocol, or single-class definition, is less than or equal to the process's maximum authorization. A process making a privileged dial_out must specify the desired access class, and it must be less than or equal to the process's maximum authorization. As described above, the channel must either be single class at the specified class, or multiclass and support set_required_access_class.

ASSURING TRUSTED PATHS

A "trusted path" is a guaranteed direct connection between a user at a terminal and the Multics operating system. The trusted path mechanism is designed to protect users against the possibility of their logging in to a simulated system created by a subverter. A trusted path should be available for all requests signalling a change in the process environment (new password, new authorization, new process, etc.).

To obtain a trusted path connection between your terminal and the Multics operating system, you must cause a terminal "hangup" condition. This will, in turn, cause the answering service (a Multics system process) to monitor the line. When you reconnect (login), you can be assured you are communicating with Multics operating system software.

To cause a "terminal hangup condition" you must cause your communications line to drop the Data Terminal Ready (DTR) signal. This can be accomplished in one of several ways:

1. Physically turning off your terminal.

2. Breaking the connection between your terminal or modem and the communications line.

3. Causing your terminal to drop the DTR signal automatically. Some terminals have this capability. (On Honeywell VIP 7800 series terminals, your place the terminal in "local" mode, and, while holding down the CTL key, typing P, D, RESET).

Once the DTR signal is dropped, you should ensure that the Multics Front-End Processor (FNP) has detected the loss of the DTR signal. It indicates this by dropping the DATA Set Ready (DSR) signal. Some terminals, modems, or multiplexers allow you to monitor this signal. You should verify that the DSR signal has been dropped. (On Honeywell VIP 7800 series terminals, the DAT Set Ready light will be go out).

Once you have verified that DSR has been inhibited, perform whatever action is necessary to reinstate the DTR signal. Several possibilities include:

1. Turning the terminal back on, if it was turned off.

2. Reconnecting the terminal or modem to the communications line, possibly redialing the phone number for Multics.

3. Causing your terminal to signal DTR again. (On Honeywell terminals, this is accomplished simply by taking the terminal out of "local" mode.

The Multics FNP will signal its receipt of DTR by reapplying DSR. This may cause the DSR indicator on your terminal or modem to light up again. the Multics Answering Service (a Multics system process) will then monitor the line, and shortly thereafter, the Multics banner will appear. When your login again, you can be assured you are communicating with the Multics operating system.

Note that this procedure assumes your communications hardware allows you to drop the DTR signal and that this loss of DTR is reflected all the way to the Multics FNP. (Some networks will prevent Multics from seeing the loss of DTR). In addition, some way of sensing the loss of DSR is required as well.

The system can be made to require a trusted path before you change authorizations (i.e., with the new_proc -auth command), or before you log out and log back in again. You can do this by setting the strict_trusted_path statement on in installation_parms with the ed_installation_parms command.

SECTION 21

ASSURING THE SECURITY OF THE I/O DAEMONS

Access to the I/O daemons is set by assigning extended access modes (adros) to the message segment established for the daemon queue. Assign access according to the following rules:

- add (a) - allows users to submit requests via this queue.
- own (o) - allows users to see and cancel their own requests. The system administrator should have r access.
- read (r) - allows users to see all requests in the queue
- status (s) - allows users to see how many requests there are in the queue. At a secure site, only the system administrator should have s access. At a moderately secure site, s access can be granted to users to allow them to see the number of requests in the queue. With s access, users cannot see requests at levels other than their current access.
- delete (d) - allows a user to delete any request in the queue. This should be set for the daemon and the system administrator only.

The suggested access that should be set is:

```
aos       *.*.*
adros     IO.SysDaemon
adros     *.SysAdmin
adros     *.SysMaint
```

GIVING A USER ACCESS TO A REQUEST TYPE

I/O daemon message queues derive their names from the Request type name specified in the I/O daemon tables. All of the I/O daemon queue message segments are located in >ddd>idd.

Give the IO.SysDaemon user identity extended access, i.e., add, delete, read, own, and status (adros) to all queues.

Use the list_acl command to list the extended access on the queues and the set_acl command to change the extended access on the queues.

GIVING A DRIVER ACCESS TO A REQUEST TYPE

Device drivers are, usually, run under the control of an IO.SysDaemon daemon process; always assign the IO.SysDaemon device driver adros extended access to daemon queues (message segments) located in >sc1>rcp.

If a daemon other than IO.SysDaemon is used as a device driver, that process must have adros extended access to the specific Request type.

The I/O daemon tables contains a driver_id statement for each request type which acts as a cross-check on the process that is supposed to run the device. IO.SysDaemon is the default daemon driver process. Similarly, for Request types you specify which drivers are allowed to run them.

ACCESS FOR CARD INPUT STATIONS

Card input stations can be either local or remote. The stations are initially defined in the I/O daemon tables. Local stations are defined with the reader_driver; remote stations are defined with the remote_driver. The args statement in the I/O daemon tables specifies the station name. Once the station name is defined in the I/O daemon tables, you must do the following to set the access for the station:

1 Create an acs for the name of the station in >scl>rcp

2. If the station is remote, register the station name in the PNT as a user name and add a password for the station.

Local stations do not require passwords; access to the local station is controlled by the acs for the station (see below). For a remote station, the acs must exist and the station name and password must be registered in the PNT. To log in the remote station, the system operator must know the station password.

To set access to the card input station acs called Station_A.acs for all users of the project Pubs, type:

```
cwd >scl>rcp
sa Station_A.acs rw *.Pubs.*
```

The method of assigning users or projects access to specific card stations protects users from remote job entry at stations they know nothing about.

ACCESS FOR ANONYMOUS BULK DATA INPUT

The segment:

```
card_input_password.acs
```

in the directory:

```
>system_control_1>rcp
```

controls the use of bulk data input by an unregistered card input user. If the user has r access to the segment, bulk data input is permitted. The user must also have access to the station access control segment.

The segment:

```
<station>.acs
```

in the directory:

```
>system_control_1>rcp
```

controls the use of a remote or local card input station by a particular user. To use the station for bulk data input the user must have r access. To use the station for remote job entry the user must have e access.

ACCESS FOR CARD INPUT DATA

Users must explicitly permit a station to submit card input for them by creating the segment "card_input.acs" in their mailbox directory:

```
>user_dir_dir>Project_id>Person_id
```

The ACL for this segment must give r access to each station that the user permits to submit bulk data input and e access for each station that the user permits to submit RJE jobs. For example:

```
re Station_A.*.*
```

The card reading process must have s access to the project_id and person_id directories. If this segment does not exist or if the access is not as specified, the card input is aborted

MARKING THE ACCESS CLASS OF DEVICES AND REQUEST TYPES

The I/O daemon incorporates certain features in support of the access isolation mechanism (AIM). System administrators at sites not using authorizations above system_low need not read the following.

Every request processed by the I/O daemon has an access class. The access class of a request is equal to the authorization of the process that submitted the request. Each piece of output normally has an access class banner. For print requests, the access class banner appears on the head sheet of each printout. For punch requests on the local punch, the access class banner appears in the flip cards at the beginning of each deck. If the access class of a request is system_low and the access class name for system_low is null, then the access class banner is omitted.

In the interest of security, some sites may find it desirable to have requests of the same type automatically separated according to access class. To illustrate how this access class separation might be used, imagine a site at which two different access classes are defined. One of these, called "public," is available to all users. The other, called "confidential," is available to only a limited number of users who deal with sensitive information. Further, suppose that the site has two printers, both of which are used to process requests of the same type. Assume that different distribution points for public and confidential output exist so that stricter control can be exercised over the release of confidential output. In this case, the operators must separate confidential output from public output by examining the access class banners. An error in bursting or separating the output could result in confidential output being accidentally released with public output. In addition to this security weakness, there is also the operational burden of separating the output according to access class.

The I/O daemon offers a solution to the above problems. Each driver process can be made to handle only requests of a single access class or a range of access classes. Therefore, all public output could be directed to one printer and all confidential output to the other. Hence, both the operational burden and the potential for operational errors mentioned above are eliminated. To facilitate output handling, the public printer could actually be located in the public distribution area while the confidential printer could be located in the confidential distribution area.

The benefits of this automatic separation are not obtained without cost. It is probable that printer utilization and hence turnaround time for output will be somewhat degraded on the whole. This is because it is unlikely that the amount of output will be evenly divided between the access classes. For example, the number of public requests might be much larger than the number of confidential requests. In this case, the confidential printer would be underutilized.

Other disadvantages become evident if one considers the situation where there are fewer printers than access classes. If instead of two printers, only one were available at the hypothetical site, then this printer would have to be switched back and forth between public and confidential output. This switching, of course, increases the operational burden. Also, it upsets the priority selection of requests. Suppose, for example, that the site decides to switch between public and confidential output every 30 minutes. A print request submitted to queue 1 might have to wait this amount of time before being processed. By contrast, if the printer were processing both access classes at once, the request would be performed immediately (assuming queue 1 were empty).

Unfortunately, even with the switching of printers from one access class to another, automatic access class separation of output simply does not scale up for a large number of access classes. Clearly, at some point it becomes impractical to rotate a small number of devices among a larger number of access classes. Therefore, sites using a large number of access classes or sites not willing to tolerate some of the drawbacks cited above may choose to forego automatic access class separation of output. In this case, each device can be made to handle the full range of access classes from system_low to system_high. Care must be taken to ensure proper distribution of output. Control forms can provide a helpful receipt for each piece of output.

At some security conscious sites, an accountability form is generated for every classified printout called a control form. This permits security control personnel to track the location of the classified printout. To do this, use the form_type keyword in the I/O daemon tables. For further information, see the *System Maintenance Procedures* manual, Order No. AM81.

Device Class

The mechanism for separating output according to access class is the "device class." Each request type can be partitioned into any number of separate device classes. One or more devices can be specified for each device class. Also, a range of access classes can be specified for each device class. When a driver process is initialized, the operator normally indicates the device to be run and the request type. However, if device classes are defined for the request type, then the operator must also indicate a device class. This determines the access class range of requests that the driver processes.

It is important to note that the device class of a request is not something the user can specify. In fact, the entire device class concept is invisible to users. Unlike the type and priority queue of a request, the device class is not determined at request submission time. Rather, it is determined at request processing time. Hence, it is possible to modify the I/O daemon tables and change the predicted device classes and this target device of requests stored in the queues.

A device class is defined by a device_class substatement in the I/O daemon tables. The device_class substatement is treated as a substatement for request types and, as such, can be freely intermixed with other substatements for request types.

device_class: <name>;

defines the name of a device class belonging to the associated request type and denotes the beginning of a device class description. Any subsequent substatements (see below) apply to this device class until the next device_class substatement or Request_type, Line, or Device statement is encountered. Any <name> may be chosen up to maximum of 24 characters.

If no device classes are explicitly defined for a request type, then a default device class is defined by implication. The primary purpose of a default device class is to allow certain substatements for device class to be specified for a request type when it has no explicit device classes. One such substatement is the device substatement that was previously described under "Substatements for Request Types." In fact, the device substatement is actually a substatement for a device class. When no device classes are defined, the device substatement applies to the default device class for the preceding request type. The same is true of all substatements for device classes.

SUBSTATEMENTS FOR DEVICE CLASSES

The substatements below describe various attributes of a device class and may appear in any order following a device_class substatement or following a Request_type statement if no device classes are defined.

min_access_class: <access_class>;
defines the minimum access class of a request to be processed in the associated device class. The <access_class> must be a standard access class string as defined in the Programmer's Reference Manual, Order No. AG91. If omitted, the default minimum is system_low.

max_access_class: <access_class>;
defines the maximum access class of a request to be processed in the associated device class. The <access_class> must be a standard access class string. If omitted, the default maximum is the access_class string given in min_access_class.

min_banner: <access_class>;
defines the minimum access class banner to be placed on the head sheet of printed output, on the flip cards of punched output, and on the control forms for all output. Normally, the access class of the request is used. However, if this access class is less than that specified for min_banner, then the min_banner value is used. The <access_class> must be a standard access_class string. If omitted, the default min_banner is the access_class string given in min_access_class.

device: <name>;
specifies a device that can be used to process requests of the associated device class. The <name> must appear as the parameter of a Device statement. More than one device substatement may be specified for a device class.

Care should be taken to ensure that the full system access range (system_low to system_high) is covered by the union of access ranges of the device classes for each request type. (If no device classes are defined for a request type, the max_access_class substatement should be set to system_high for the default device class.) If not, requests of access classes that are not included are never processed. Upon discovering such a request, the I/O coordinator prints an error message and skips the request. Also, it should be noted that if two or more device classes from the same request type have overlapping access ranges, then a request falling in this overlap is assigned to the device class defined first in the I/O daemon tables source file (iod_tables.iodt).

As mentioned above, when multiple device classes are defined for a request type, requests are generally not performed in the usual order dictated by priority and submission time. This phenomenon is most noticeable when one device must be shared among several device classes. In order to aid the operators in determining when to switch a device to a different device class, the I/O coordinator keeps track of "waiting" requests. A waiting request is one that is passed over in the normal request selection order while the coordinator looks for a request to satisfy a different device class, or is explicitly requested to run at high priority by an operator command. A count of waiting requests is kept on a per device class basis. When the number of waiting requests for a device class becomes large, this indicates that the device class is receiving inferior service relative to some other device class for the same request type. Thus, operators could be instructed to switch a device to another device class whenever the number of waiting requests reaches some limit.

SUBSTATEMENT FOR DEFAULT REQUEST TYPE

The default_type substatement described earlier under "Substatements for Devices" names the default request type that a device processes unless overridden by the operator. However, if device classes are defined for the request type, then the parameter of the default_type substatement must include both the request type and device class names separated by a period (e.g., printer.confidential).

SOURCE FILE EXAMPLE USING AIM

This example shows a portion of a source file that illustrates the use of AIM features.

```
Request_type:          printer;
  generic_type:        printer;

device_class:          public;          /* for system_low output */
  device:              printer_1;       /* primary public printer */
  device:              printer_2;       /* can use this one in
                                           emergencies*/

device_class:          confidential; /* for output above system_low */
  min_access_class:    level1;
  max_access_class:    system_high;
  min_banner:          level2;          /* all confidential output
                                           has at least a level 2 banner
                                           authorization */

  device:              printer_2;       /*  use only this printer located
                                           in secure area */

Device:                printer_1;
  driver_module:       printer_driver_;
  prph:                prta;
  default_type:        printer.public;

Device:                printer_2;
  driver_module:       printer_driver_;
  prph:                prtb;
  default_type:        printer.confidential;

Request_type:          punch;
  generic_type:        punch;
  max_access_class:    system_high;     /* handle all access classes */

Device:                punch_1;
  driver_module:       punch_driver_;
  prph:                puna;
  default_type:        punch;
```

SECURING PRIVILEGED DAEMONS

Daemon processes are used for several privileged applications. The hierarchy backup, volume backup, salvager, scavenger, and utility daemons are the primary examples. These daemons require access to functions that should not be available to operators. However, by using the message coordinator quit and reply command, an operator can take complete control of any of them. Therefore, to secure the daemons from unauthorized access, special ACS segments must be created to control access to the daemons.

To check the access to a daemon, special access names are used to check access to the message coordinator. The two conventions that are used are:

- First, if an identified and authenticated user_id is not available, then a special name beginning with "_" is used for the Person component of the access name.

- Second, the "Operator" project, already documented for ordinary login by operators, will be used as the Project component for operators logged in via the "sign_on" initializer command, and the new tag "o" will be used for the Tag.

The following list describes all the sources of daemon control commands and the access name that are used when checking access:

- Ordinary Operator Commands - An operator typing the reply or quit command on an initializer terminal or the bootload console. If the operator is not logged in, then the special access name "_Unidentified.Operator.o" is used. Note the use of the special person_id "_Unidentified". If the operator is logged in, then "Person.Operator.o" is used. This permits selective control based on the individual operator.

- Exec commands - An operator typing an "exec" (x) command on an initializer terminal or the bootload console. The admin ec, or another ec that it calls, uses sc_command to execute a daemon control command. In this case, the access name "_Exec_Command.Operator.o" is used. Note the use of the special person_id "_Exec_command". A special access name here reflects the fact that the admin ec is built by trusted staff, and can be assumed to only permit the operators to do a suitably restricted set of things.

- sc_commands in admin mode - An operator has entered admin mode, and used sc_command to enter a daemon control command. In this case, the access name is "Initializer.SysDaemon.z", since admin mode is equivalent to full control of the system.

  send_admin_command sc_commands - A privileged user has send an admin command. In this case, the privileged user's access name is used verbatim.

These access names allow administrators to express precise distinctions. You should deny access to "*.Operator.o", and then grant it to "_Exec_Command.Operator.o". The necessary functions would be put in the admin exec_com, and operators would not be required to type Multics commands to daemons.

An extended object ACS segment is used to hold the ACL for each message coordinator source. The extended object model is used because the standard segment "rew" bits cannot be used to describe the different access modes that must be defined for a message coordinator source. Since these modes (defined below) are unique to the message coordinator, the unique suffix ".mcacs" is used instead of ".acs."

To determine the effective access of the access name to the daemon source, the Message Coordinator determines the extended ACL of the segment >sc1>mc_acs>SOURCE_NAME.mcacs. For example, if the Utility daemon is logged in on source "ut", then the ACS would be >sc1>mc_acs>ut.mcacs.

Suffix_mcacs_ defines the following ACL modes:

- r - REPLY access, permits the user to reply to the daemon.
- q - QUIT access, permits the user to send a quit to the daemon.
- c - CONTROL access, permits the user to log the daemon in and out.
- d - DAEMON access, permits a DAEMON to log in on this source. This mode controls the association of daemon names with message coordinator sources. For an operator to log Daemon A in on source B, the operator must have c access on the source and the Daemon must have l access.

Implementing Privileged Daemon Security

The acct_start_up.ec creates the mc_acs directory, and puts ACS segments in it for all the daemons manipulated in the standard admin.ec. For the Salvager, Scavenger, backup, and Utility daemons, the acct_start_up.ec sets the ACL to:

```
crq   Initializer.SysDaemon.z
crq   _Exec_Command.Operator.o
crq   *.SysMaint
crq   *.SysAdmin
cdrq  Backup.SysDaemon.z (or whatever daemon is to log in here)
null  *.*.*
```

The IO daemon, who has a limited command environment, will give rcq to *.*.o.

When the "validate_daemon_commands" parameter is set to "on" in ed_installations_parms, the reply, quit, login, and logout commands will:

- Require that the segment >sc1>mc_acs>SOURCE.mcacs exist before logging in a daemon on the source.
- Check the extended ACL on the mcacs segment for the access of the special user name defined above, and forbid the command if the required access is missing. Demand that the Daemon to be logged in have "d" effective access to the segment.

The validate_daemon_commands should be set "on" at a secure site.

SECTION 22

ABSENTEE FACILITY SECURITY

Absentee card-input job security lets you control which users have access to specific absentee queues by manipulating the access control segment of the absentee queue in the >sc1 directory, and which daemon processes have access to Absentee driver process by manipulating the access to the >sc1>admin_acs>absentee_proxy.acs segment.

GIVING A USER ACCESS TO AN ABSENTEE QUEUE

There are five absentee queue segments that you grant some or all users access to, as follows:

- >sc1>absentee_foreground.ms
- >sc1>absentee_1.ms
- >sc1>absentee_2.ms
- >sc1>absentee_3.ms
- >sc1>absentee_4.ms

Users add requests to these queues to run absentee jobs in the foreground queue and background queues 1, 2, 3, and 4 respectively.

To allow users to submit requests in a particular queue, give them ao access. For example:

```
sa >sc1>absentee_1.ms ao *.HooverDam.*
```

This allows them to add requests and to see and delete requests that they added.

To allow users to find out how many requests are in the queue, give them "s" access. For example:

```
sa >sc1>absentee_3.ms aos *.BusyBody.*
```

To allow users to see the contents of each other's requests, give them "r" access. For example:

```
sa >sc1>absentee_4.ms aros *.Snoop.*
```

In general, all users are given aos access to all queues to which they are permitted to submit requests. *.SysAdmin, *.SysMaint, and *.SysDaemon are given adros to permit them to delete other users' requests. At open sites, users are given aros access to the queues.

GIVING A DAEMON ACCESS TO PROXY

The segment absentee_proxy.acs in the directory >sc1>absentee_acs controls the use of the proxy absentee facility. Any user with e access to this segment can submit proxy absentee jobs, that is, jobs to be run on behalf of, and under the User_id of, other registered users. Normally only daemons that operate the GCOS batch facility or the remote job entry (RJE) facility and the SysAdmin project should be given e access to this segment. Any other project or user with access to absentee_proxy.acs could cause security breaches.

1. cwd >sc1>admin_acs

2. sa absentee_proxy.acs rew RJE.SysDaemon.z

SECTION 23

PRIVILEGED OPERATIONS SECURITY

UNDERSTANDING PRIVILEGE ON MULTICS

"Privilege" means to have special access rights to various aspects of the system. A privilege is an attribute of a process that permits it to perform operations that are denied by normal access rules. In other words, privileged access means that the system will grant you the right to perform some operation that is prohibited by normal site policy. There are two general categories of privilege:

- The normal model of administrative access that permits you to access certain data bases to register projects and users, manipulate accounting data bases, or any of the other system administrative functions that require special access. This is also called "admin access".

- Special purpose access that is out of the normal model of administrative access. This type of access permits administrators to repair, for example, damage to the file system, or circumvent AIM restrictions. These types of procedures are not normally part of administrative access rights.

Note that admin access is within the restrictions imposed by AIM and ACLs; special privilege is not.

As system administrator, you are a person who, under the normal model of administrative access, has the right to perform actions that very few people have the right to perform; in addition, you have the ability to circumvent the access rules established by the normal administrative model.

Access to circumvent the normal administrative model is granted by assigning access to special privileged gates (see "Access to Privileged Gates", later in this section).

The following aspects of system privilege are discussed below:

- granting access to use system privileges
- using system privileges
- setting access to access control segments for system table installations
- setting access for large I/O buffers
- Setting access to privileged gates

GRANTING ACCESS TO USE SYSTEM PRIVILEGES

System privileges refer to granting specific individuals the right to circumvent AIM restrictions. Circumventing the system policies for AIM is a very serious responsibility. There is no specific command or gate that grants you full immunity from AIM restrictions. However, certain gates permit you to selectively give yourself the right to circumvent AIM.

For example, AIM dictates that you cannot write into a segment that is at an access class lower than yours and that you cannot read a segment at a higher access class. If you have access to the system_privilege_ gate (see below) you can use the set_system_priv command (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64) to temporarily give yourself the right to circumvent those rules.

Circumventing any AIM restrictions causes an entry to be placed in the system log indicating that AIM privileges were circumvented and the person who circumvented them. This record of AIM access circumvention provides an extra measure of security that permits you and other administrative personnel be aware of any AIM circumvention.

Once you have completed the required actions on the segments, you can then reset the AIM restrictions using the set_system_priv command.

USING SYSTEM PRIVILEGES

The set_system_priv command lets you to turn on and off the system privileges that allow the process to function outside the restrictions of the AIM access controls. You must have access to the system_privilege_ gate to use this command. Whenever the privileges are turned on, you must carefully check to ensure that your actions do not accidentally disclose information that was previously protected by the AIM access controls. The set_system_priv command is needed only if the site is using the AIM access controls.

The list of system privileges that can be enabled with the set_system_priv command are:

- comm - turn on communications privilege.
- dir - turn on directory privilege.
- ipc - turn on interprocess communication (IPC) send/receive privilege.
- rcp - turn on privilege for RCP resource management.
- ring1 - turn on privilege for ring 1 subsystems.
- seg - turn on segment privilege.
- soos - turn on security-out-of-service privilege.

Note that it is important to turn the system privileges off after you have used them. If system privileges are not turned off after they are used, serious security breaches could result.

You must be especially careful to avoid Trojan horses when using privileges. Do not have people's private libraries in your search rules. You can reduce your exposure by logging into ring 2 or ring 3 while using privileges, so that you will take outward_call errors if you try to use code that is not installed in the system libraries.

SETTING ACCESS TO THE ACS FOR SYSTEM TABLE INSTALLATIONS

In the directory >sc1>admin_acs, there is one access control segment per installable system table. Individuals with rw access to the table ACS segments are permitted to install them. The installable system tables are:

- ttt.install.acs
- mst.install.acs
- sat.install.acs
- rtdt.install.acs
- cdt.install.acs

The administrative tables which include the PNT, the installation_parms in >sc1, and other project registration files are all directly manipulated by system administrators. Access to these data bases is set with the set_acl command and they do not have corresponding acs entries in >sc1>admin_acs.

SETTING ACCESS TO THE ACS FOR LARGE I/O BUFFERS

Large I/O buffers represent a class of miscellaneous things that are controlled by ACS segments. Access to the segment >sc1>rcp>workspace.acs gives a user the right to commit more memory than usual for them when they are doing I/O. It is necessary for a user to have this privilege if the user is doing tape I/O, for example, and using very large tape I/O buffers. In general, only those users requiring this feature should have access to the workspace.acs segment.

To permit a user to use large workspaces, type:

```
cwd >sc1>rcp

sa workspace.acs rw Person_id
```

SETTING ACCESS TO PRIVILEGED GATES

The following is a list of privileged gates located in the >system_library_1 directory:

- access_audit_gate
- dm_hphcs_
- hc_backup_
- initializer_gate_
- mhcs_
- phcs_
- shcs_
- system_privilege_
- tandd_
- hphcs_
- initializer_mdc_
- mdc_priv_
- rcp_priv_
- rcp_sys_
- rcp_admin_

The following is a list of privileged gates located in the >sss directory:

- dm_admin_gate_
- dm_daemon_gate
- mail_table_initializer_
- mail_table_priv_
- metering_gate_
- queue_admin_
- rl_io_
- user_message_admin_
- user_message_priv_

The following is a list of privileged gates located in the >t directory:

- installation_tools_
- pnt_admin_gate_
- pnt_login_gate_
- pnt_network_gate_
- pnt_priv_gate

The following is a list of privileged gates located in the directory >unb:

- forum_admin_gate_
- forum_chairman_

Unless otherwise stated, set access to privileged gates by adding the appropriate entries to the system_start_up.ec file using the hpsa command. For daemon processes, access to the gates below is assigned automatically as required. It is your responsibility to determine which individuals will have access to specific gates depending on the site security policy.

The following example illustrates the general procedure for setting access to privileged gates in the system_start_up.ec file using the hpsa command. To use the hpsa command, you must have access to the hpsa_ gate (see below):

```
hpsa gate_name access_mode Person_id
```

The access_audit_gate_

The >system_library_1>access_audit_gate_ is a highly privileged gate for use by elements of the ring-4 TCB (e.g. printer daemon). The recommended access for the >system_library_1>access_audit_gate_ is:

```
re          *.SysAdmin.*
re          *.SysDaemon.*
re          *.SysMaint.*
```

The dm_hphcs_ Gate

The >system_library_1>dm_hphcs_ gate is a highly privileged gate used by Data_Management.Daemon to call from ring 2 to ring 0 to invoke hardcore support routines. Data_Management.Daemon has re access to this gate.

The recommended access for the >system_library_1>dm_hphcs_ gate is:

```
re          Data_Management.Daemon.*
re          *.SysDaemon.*
```

The hc_backup_ Gate

The >system_library_1>hc_backup_ gate allows volume backup daemons to access the file system without regard for access control. It also has specialized entries to aid the volume backup daemons in accomplishing their task. The recommended access for the >system_library_1>hc_backup_ gate is:

```
re          Volume_Dumper.Daemon.*
re          Volume_Reloader.Daemon.*
re          Volume_Retriever.Daemon.*
re          *.SysDaemon.*
re          *.SysAdmin.*
re          *.SysMaint.*
```

The initializer_gate_

The >system_library_1>initializer_gate_ is a very privileged gate for manipulating the storage system and system parameters. Only Initializer should have access to this gate. The recommended access to the >system_library_1>initializer_gate_ is:

```
re          Initializer.SysDaemon.z
```

The mhcs_ Gate

Security conscious sites that are concerned with the use of covert channels should restrict the use of this gate to the following:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
```

The phcs_ Gate

Access to the >system_library_1>phcs_ gate lets you to look at anything in the system. It does enforce ACLs on segments but it also lets you read the contents of objects that are only ring
* protected and have ACLs that permits read access. It also allows you to read the contents of any user mailbox.

Access to the phcs_ gate should be assigned very sparingly according to your site security policy. There is no conventional operation which requires access to the phcs_ gate, but system maintainers use it to view directories if a problem arises.

Note that users with access to phcs_ can completely compromise system security; it should not be given out to untrusted individuals.

The recommended access to the >system_library_1>phcs_ gate is:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
re        *.Daemon.*
re        *.HFED.*
```

The shcs_ Gate

The >system_library_1>shcs_ gate allows setting the limit on the number of forced disk writes allowed. The recommended access to the >system_library_1>shcs_ gate is:

```
re        *.SysDaemon.*
re        *.SysMaint.*
re        *.SysAdmin.*
```

The system_privilege_ gate

The sole purpose of the >system_library_1>system_privilege_ gate is to permit circumvention of AIM restrictions. Access to system_privilege_ should only be given to system security administrators and/or people who have to repair file system damage that is AIM related.

You need access to system_privilege_, for example, when there is a piece of the file system that is violating AIM rules due to hierarchy damage. An example of this would be no terminal quota on a directory that is upgraded in AIM classification, or there is a directory that is below the access class of its containing directory.

On a site that is using AIM, access to this gate should be given very sparingly since access to it allows a user to violate all AIM restrictions.

Since a system always runs AIM, even if the access categories and classes are not differentiated, an object might incur an AIM classification by accident. In this case, access to system_privilege_ can be given to system maintainers who might have to use it to remove the incorrect AIM marking on a segment.

The recommended access to the >system_library_1>system_privilege_ gate is:

```
re        *.SysDaemon.*
re        *.SysMaint.*
re        *.SysAdmin.*
```

The tandd_ Gate

The >system_library_1>tandd_ gate provides various entrypoints required for running test and diagnostics. The recommended access to the >system_library_1>tandd_ gate is:

```
re        *.SysDaemon.*
re        *.SysMaint.*
re        *.SysAdmin.*
re        *.HFED.*
```

The hphcs_ Gate

Access to the >system_library_1>hphcs_ gate lets you assign yourself access to any system function and circumvent all system access restrictions. The appropriate security policy for the hphcs_ gate is to give no one access to it except *.SysDaemon and *.SysAdmin. If anyone else requires access to hphcs_ they must use the send_admin_command command to give themselves access to hphcs_ if they have some temporary need for access to hphcs_. The hphcs_ gate lets you patch ring 0 arbitrarily.

The recommended access for the >system_library_1>hphcs_ gate is:

```
re    Data_Management.Daemon.*
re    *.SysDaemon.*
```

The initializer_mdc_ Gate

The >system_library_1>initializer_mdc_ gate is the privileged master directory control and logical volume control gate for the Initializer only. The recommended access to the >system_library_1>initializer_mcd_ gate is:

```
re        Initializer.SysDaemon.z
```

The mdc_priv_ Gate

The >system_library_1>mdc_priv_ gate lets you perform a variety of administrative operations to the logical volume system. The recommended access to the >system_library_1>mdc_priv_ gate is:

```
re    *.SysDaemon
re    *.SysAdmin.*
re    *.SysMaint.*
```

The rcp_priv_ Gate

The rcp_priv_ gate contains RCP circumvention entry points. Access to it should only be given to individuals running Test and Diagnostic programs. This is, basically, the sole purpose of the rcp_priv_ gate.

An example of a use for rcp_priv_ would be if you wanted to mount a tape on a tape drive that was too damaged to even ask to look at the tape label. Access to rcp_priv_ allows you to circumvent the label checking security.

The rcp_priv_ gate also allows you create attachments to tape controllers or disk controllers that, normally, cannot be attached.

The recommended access to the >system_library_1>rcp_priv_ gate is:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
re        *.Daemon.*
re        *.HFED.*
```

The rcp_sys_ Gate

Access to the >system_library_1>rcp_sys_ gate permits you to manipulate peripherals that belong to the system by way of system processes such as:

- Volume_Dumper.SysDaemon
- Volume_Retriever.SysDaemon

These system processes have the ability to do direct I/O to disk packs that appear to be Multics storage system volumes. Note that users are not permitted to write to Multics storage system volumes.

For example, if a user attempts to load a disk pack and Multics detects a storage system label, Multics will not allow the disk pack to be mounted. The only processes permitted to manipulate a disk pack with a Multics storage system label on it are those with access to rcp_sys_.

The rcp_sys_ gate also permits processes with access to it to reserve devices for system use (make system attachments).

Access to the rcp_sys_ gate should be assigned to the system processes listed above and individuals who do operations like system retrievals and system reloads.

The recommended access to the >system_library_1>rcp_sys_ gate is:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
re        *.HFED
re        Volume_Dumper.Daemon.*
re        Volume_Reloader.Daemon.*
re        Volume_Retriever.Daemon.*
```

The rcp_admin_ Gate

The >system_library_1>rcp_admin_ gate allows you to be a RCP administrator. This gate is used only in conjunction with RCPRM. It gives you access to register and deregister volumes and devices.

An individual with access to rcp_admin_ in addition to being able to create and destroy objects, can:

- Acquire a tape on someone's behalf
- Take back a tape that was assigned to another user
- Release a "release locked" resource. A release locked resource, at an AIM site, is not returned for general use when it is released by a user; it is kept separate until it has been erased.
- Use the -priv with most of the RCPRM commands.

The recommended access to the >system_library_1>rcp_admin_ gate is:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
```

The dm_admin_gate_

The >sss>dm_admin_gate_ gate is a privileged gate that grants administrative powers over the Data Management environment. Those with the proper access can schedule DMS shutdown, view the status of all transactions, send special requests to the DM daemon, and otherwise perform all the functions and services provided by the privileged data management commands described in the *Administration, Maintenance, and Opeations Commands* manual. Order No. GB64. Data_Management.Daemon and .SysAdmin have re access to dm_admin_gate_.

The recommended access to the >sss>dm_admin_gate_ is:

```
re          *.SysDaemon.*
re          *.SysAdmin.*
re          *.SysMaint.*
re          Data_Management.Daemon.*
```

The dm_daemon_gate_

The >sss>dm_daemon_gate_ gate is a highly privileged gate that enables the DM daemon to serve as caretaker of the Data Management environment in performing such tasks as adjusting transactions on rollback and cleaning up dead processes. Access to this gate is required to kill a transaction. Data_Management.Daemon has re access to this gate.

The recommended access to the >sss>dm_daemon_gate_ is:

```
re          *.SysDaemon.*
re          *.SysAdmin.*
re          *.SysMaint.*
re          Data_Management.Daemon.*
```

The mail_table_initializer_ Gate

The >sss>mail_table_initializer_ gate is used by the Initializer process to update the mail table when users change their default project at login time. The recommended access to the >sss>mail_table_initializer_ gate is:

```
re          *.SysDaemon.*
```

The mail_table_priv_ Gate

The >sss>mail_table_priv_ gate permits manipulation of a table containing system defined mail destinations. The recommended access to the >sss>mail_table_priv_ gate is:

```
re          *.SysDaemon.*
re          *.SysAdmin.*
re          *.SysMaint.*
```

The metering_gate_

The >sss>metering_gate_ gate lets you view a set of system metering values in ring 0 that do not compromise security. Access to the metering_ gate does not permit you to read user data; however, it might be possible to establish a signalling path across AIM boundaries. Access to metering_ gate allows individuals to view system metering data which is not, generally, a security issue. However, because of covert channels, access to this gate should be restricted.

The purpose of metering_gate_ is to allow an individual to see system meters without giving them the right to see anything at all in ring 0 which might include user terminal buffers, or passwords or any other sensitive data that is kept in ring 0.

The recommended access to the >sss>metering_gate_ is:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
re        *.HFED.*
```

The queue_admin_ Gate

The >sss>queue_admin_ gate allows arbitrary manipulation of ring-1 message segments. This is a critical gate. Access to this gate allows a user to submit arbitrary absentee requests and manipulate queue entries for anyone. The recommended access to the >sss>queue_admin_ gate is:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
re        IMFT.Daemon.*
```

The r1_io_ Gate

The >sss>r1_io_ gate allows cross-ring I/O between ring-1 and the user-ring. Since there may be problems with covert channels and parameter validating, it is best not to allow access to this gate to anything other than trusted TCB processes. The recommended access to the >sss>r1_io_ gate is:

```
r      *.SysDaemon.*
```

The user_message_admin_ Gate

The >sss>user_message_admin_ gate allows privileged administrative processes to examine the messages being held by the privileged user messages facility. The recommended access to the >sss>user_message_admin_ gate is:

```
re        *.SysDaemon.*
re        *.SysAdmin.*
re        *.SysMaint.*
```

The user_message_priv_ Gate

The >sss>user_message_priv_ gate allows privileged processes to send messages (not normal messages) to groups of processes or individual processes. The recommended access to the >sss>user_message_priv_ gate is:

```
re      Initializer.SysDaemon.z
r       *.SysDaemon.*
```

The installation_tools_ Gate

The >tools>installation_tools_ gate lets you manipulate the file system as if you were in ring 1. Its intent is to allow selected individuals to manipulate system libraries which, by default, are kept in ring 1. It also gives you access to system mailboxes and queues. However, installation_tools_ respects ACLs and AIM. This means that you can only manipulate segments in ring 1 to which you have ACL access. The installation_tools gate does allow access to ring 1 and an individual with access to installation_tools_ can look at anything in ring 1 to which he has access and this includes other people's mailboxes.

Installation_tools_ resides in >tools. Access to it is maintained with ACLs set on the segment.

The installation_tools_ gate can also be used to give access to ring 2 or ring 3 to those individuals or projects that require it. This is done by copying the source of the installation_tools_ gate and installing it in ring 2 or ring 3.

The recommended access to the >tools>installation_tools_ gate is:

```
re      *.SysDaemon.*
re      *.SysAdmin.*
re      *.SysMaint.*
```

The pnt_admin_gate_

The >tools>pnt_admin_gate_ is used for System Administrative functions. All people who are allowed to read or write PNT entries need access to this gate. No passwords can be retrieved using the entries in this gate since passwords are one-way encrypted. Access to this gate permits reading of PNT entries including the encrypted password.

The recommended access to the >tools>pnt_admin_gate_ is:

```
re          *.SysDaemon.*
re          *.SysAdmin.*
```

The pnt_login_gate_

The >tools>pnt_login_gate_ is used for identification and authentication upon login. The recommended access to the >tools>pnt_login_gate_ is:

```
re          Initializer.SysDaemon.z
r           *.SysDaemon.*
```

The pnt_network_gate_

The >tools>pnt_network_gate_ is used for IMFT, Card_Input, and IO.SysDaemon manipulations of "network" (formerly "card-input") passwords. All userids which manipulate and validate network passwords need re access to this gate. This includes the IMFT daemons, and any daemons which manipulate card input.

The recommended access to the >tools>pnt_network_ gate is:

```
re          *.SysDaemon.*
re          IMFT.Daemon.*
re          Card_Input.Daemon.*
```

The pnt_priv_gate_

The >tools>pnt_priv_gate_ allows access to passwords. Not used at this time, but available for very privileged applications which must perform statistical analysis of passwords. Access to this gate should not be handled out except to the most trusted of system administrators, and only if absolutely necessary. There is no need to have any access on this gate.

The recommended access to the >tools>pnt_priv_gate is:

```
r           *.SysDaemon.*
re          *.SysAdmin.*
```

Forum Gates

The Forum Interactive Meeting Facility, if present on your site, has two privileged gates:

- forum_chairman_ - All users should have access to it. It allows users to create a new forum and be chairman of it.
- forum_admin_ - Permits circumvention of forum ACLs and perform normally restricted operations on all forums. It also lets you use the forum_admin command.

You can restrict forum proliferation and usage by setting selective access to the forum_chairman_ gate but, generally, access to this gate is not restricted.

Access to forum_admin_ should be given to system maintainers who are responsible for maintaining Forum.

SECTION 24

SYSTEM LOGS

While the system is running, several kinds of information may be generated describing the normal operation of the system or recording abnormal occurrences. The information is generated in several different ways and is stored in the Multics storage system in a segment or series of segments called logs so that it can be processed by Multics programs or special logging commands. These logs are:

- syserr log
- answering service log
- admin log
- message coordinator logs
- data management log

Each of the above logs can be manipulated with the following logging commands:

- print_sys_log - prints messages from a log
- monitor_sys_log - monitors traffic in a log
- summarize_sys_log - produces a summary of log contents
- move_log_segments - moves log segments from one place in the hierarchy to another for archival storage

A description of each system log is given below.

The Syserr Log

The syserr log segments are located in >sc1>syserr_log. Messages reporting system-detected errors, device usage, hardware errors and status, maintenance events, security related events, and other significant events are recorded by the supervisory software in the syserr log.

The syserr log actually consists of three stages: a small wired buffer where messages are queued, a larger paged buffer (the LOG partition) where messages are held until they can be copied into the hierarchy, and a permanent log in >sc1>syserr_log, where they remain as long as you deem useful. This mechanism is transparent during normal operation; the log manipulation tools present the log messages in the proper sequence.

To print the contents of the syserr_log, use the following command:

```
print_sys_log -syserr
```

For additional information on the syserr_log, see Section 25.

The Answering Service Log

The answering service log is located in >sc1>as_logs. The answering service log is used by the Answering Service (running in the Initializer process) to record events (errors and nonerrors) of interest to the site. Any errors encountered in the process of doing Answering Service business are recorded in the answering service log. In addition, the answering service log includes accounting information, identification and authentication information (logins and logouts), process creation/destruction information, communications channel attachments/detachments, and dial service information are all kept in the answering service log (>sc1>as_logs>log).

All commands given to the answering service are recorded in the answering service log including operator input to the system and the PDT/CDT/SAT/RTDT table installations.

Answering service logs are written only by the Initializer; applications should not attempt to write to the answering service log or any message coordinator logs.

Note that since the answering service log contains information about aborted login attempts, mistyped user passwords may be identifiable in the log. It is important that you safeguard the answering log from unauthorized access.

To print the contents of the answering service log, use the print_sys_log command as shown below.

```
print_sys_log -as
```

For additional information on the answering_service logs, see Section 25.

The Admin Log

The admin log is located in >sc1>as_logs. The admin log records use of admin mode and send_admin_command commands.

To print the contents of the admin log, use the print_sys_log command as shown below.

```
print_sys_log -admin
```

For additional information on the admin log, see Section 25.

The Message Coordinator Logs

Message coordinator logs are located in >sc1>as_logs. The iolog is a system defined message coordinator log and is supplied with the system. You can create other message coordinator logs that record events that you want to audit by using the define Initializer command and the route command. For every log written using the define command, a log is created. For detailed information on creating message coordinator logs. refer to the define command and the route command *(see the Administration, Maintenance and Operations Commands manual,* Order No. GB64).

In the example below, a define command and a route command (placed in the system_start_up.ec) are used to route all output from a printer named prta into a log called the iolog located in >sc1>as_logs.

```
sc_command  define iolog log iolog
sc_command route prta user_i/o iolog
```

To print the information contained in the iolog, use the following command:

```
print_sys_log -mc_log iolog
```

Data Management Log

The Data Management log is located in >site>dm>system_low>logs>dm_system_log. The Data Management log records messages during:

- Data Management initialization or startup
- Normal system operation
- Data Management shutdown

Data management initialization or startup messages indicate the status of Data Management since the last Data Management initialization. The following are examples of Data Management initialization messages.

```
14:46:14 1000000 4 (Newcomb.Multics) ENTERING system crash recovery
                   code.
14:46:36 1000001 4 (Newcomb.Multics) Directory to recover:
                   >udd>m>cp>lan>dms.test.sys_dir>system_low>dm_dir.4112071328.
14:46:50 1000002 4 (Newcomb.Multics) Number of journals opened for
                   crash recovery = 1.
14:47:00 1000003  4 (Newcomb.Multics) :
                   1 txns to recover in
                   >udd>m>cp>lan>dms.test.sys_dir>system_low>system_default.bj.
14:47:26 1000004  4 (Newcomb.Multics) :
                   No errors recovering before journal
                   >udd>m>cp>lan>dms.test.sys_dir>system_low>system_default.bj
14:47:31 1000005 4 (Newcomb.Multics) FINISHED recovery of
                   >udd>m>cp>lan>dms.test.sys_dir>system_low>dm_dir.4112071328.
14:47:33 1000006  4 (Newcomb.Multics) bjm_per_system_init$part_2
14:47:45 1000007 3 (Newcomb.Multics) DMS shutdown scheduled - User
                   warning at 04/02/85 2145.0 est Tue, begin shutdown
                   at 04/02/85 2150.0 est Tue, user shutdown at
                   04/02/85 2155.0 est Tue, users bumped at 04/02/85
                   2200.0 est Tue, daemon logged out at 04/02/85
                   2205.0 est Tue.
14:47:48 1000008  3 (Newcomb.Multics) Data Management System initialized.
```

Normal Data Management operation messages include:

- idle timeout messages
- abandoned transaction messages
- killed transaction messages

Idle timeout messages are logged when no activity has occurred in the data management system for a specific time interval. The Data Management system "wakes up" on a periodic basis to determine if any cleanup activity is required and produces a message in the log indicating the wakeup intervals and the action taken.

The following is an example of a Data Management idle timeout message.

```
08:51:18 1001495  4 (Data_Management.Daemon) Idle timeout request by
                    004200003701, (Data_Management.Daemon.z).
```

Abandoned transaction messages are logged when a user abandons a transaction. The following are examples of abandoned transaction messages:

```
13:51:31 1001578  3 (Hanley.Nomad) User abandoned transaction 2, TID = 4,
                    pid = 011700007702, owner = Spratt, state = 72
13:51:31 1001579  4 (Data_Management.Daemon) ADJUST_TXN (index = 2) requested
                    by 011700007702 (Hanley.Nomad.a).
13:51:33 1001586  4 (Data_Management.Daemon) Transaction successfully adjusted.
```

Killed transaction messages are logged when a system administrator kills a specific transaction. Only system administrators are allowed to "kill" a transaction. The following are examples of killed transaction messages.

```
16:24:10 1001613  4 (Data_Management.Daemon) KILL_TXN (index = 5) requested by
                    026113656020 (Dunbar.SysProj.a).
16:24:11 1001614  3 (Data_Management.Daemon) User requested kill, transaction
                    5, TID = 12, pid = 02367421, owner = Dunbar, state = 56
```

Data management shutdown messages are logged at Data Management shutdown. The following are examples of Data Management shutdown messages.

```
19:49:35 1001667  4 (Data_Management.Daemon) SHUTDOWN requested by
                    014155260237 (Schred.SysAdmin.a)
19:49:38 1001668  3 (Data_Management.Daemon) DMS shutdown scheduled - User
                    warning at 03/25/85 1945.0 mst Mon, begin shutdown at
                    03/25/85 1950.0 mst Mon, user shutdown at 03/25/85
                    1955.0 mst Mon, users bumped at 03/25/85 2000.0 mst Mon,
                    daemon logged out at 03/25/85 2015.0 mst Mon.
19:50:01 1001715  3 (Data_Management.Daemon) Turning off transaction begins.
```

To print the information contained in the Data Management log, use the following command:

```
print_sys_log -dms
```

You can use other control arguments in conjunction with the -dms control argument to print selected portions of the Data Management log.

SETTING ACCESS TO LOGS

Logs reside in their respective directories, as described above. The initial access on the containing directory determines the access that is set for the logs created in that directory. To set the initial access on the containing directory for the appropriate logs, use the set_iacl_segs (sis) command (see the *Commands and Active Functions* manual, Order No. AG92).

Since the initial ACL is applied to the log segments only as they are created, it is necessary to set access on the existing log segments, you can do this with the standard set_acl (sa) command.

The initial access on >sc1>syserr_logs and >sc1>as_logs should give the process that runs the crank rw (read write) access to all segments and sma (status, modify, append) to the log directory. This permits the crank to correct log segment headers when moving old log segments into >udd>sa>a>history (for a description of history logs, see below).

To set the access for all logs residing in the directory >sc1>as_logs, use the command:

```
set_iacl_segs >sc1>as_logs r .Multics.; sa >sc1>as_logs>* r .Multics.*
```

This command gives all users of the Multics project access to read the logs created in >sc1>as_logs.

HISTORY LOG FILE

The crank moves old log segments from the >sc1>syserr_log and >sc1>as_logs directories into the >udd>SysAdmin>history directory. This is done using the move_log_segments command which preserves proper ordering of the logs. The log perusal tools (e.g., print_sys_log) can retrieve messages from the old log segments without special usage requirements.

When using the move_log_segments command, it is important to follow the correct procedure. Consider the following example.

After moving log segments (i.e., admin_log or log or iolog) out of one directory (i.e., >udd>sa>a7h) into another directory (udd>sa>old_logs) using the move_log_segments command, you want to move the logs to a third directory (udd>sa>old_iologs) and still have the commands find the logs in the new directory.

Renaming the old_logs directory will not work, since the pathname >udd>sa>old_logs is recorded in the log header of the oldest log in >udd>sa>a>h, and since old_logs may also contain other log families that you want to leave in that directory.

In the header of each log segment is the pathname of the directory in which the next older log segment is to be found; this is called the log_history_dir.

The age of a log segment (relative to other segments of the log family) is NOT DETERMINED by the segment's date/time-contents-modified value. It is determined by the entry name of the log segment; i.e., log.19861119.030533. The last two components represent the time stamp of the log in GMT: "YYYYMMDD.HHMMSS". This avoids problems when switching between daylight-standard and daylight-savings time in the spring and fall.

The diagram below shows how the logs in a log family are chained together using the log_history_dir in the header of each log segment.

LOG DIRECTORY	LOG SEGMENT	LOG HISTORY DIR
>sc1>as_logs	iolog	>sc1>as_logs
.....	iolog.19861119.063822	>udd>sa>a>h
>udd>sa>a>h	iolog.19861117.223517	>udd>sa>a>h
.....	iolog.19861115.093546	>udd>sa>old_logs
>udd>sa>old_logs	iolog.19861113.074657	>udd>sa>old_logs
.....	iolog.19861111.151709	(null string)

The log_history_dir is changed as necessary by the move_log_segments command, normally in the system's daily crank absentee. When the command moves a log segment to a new directory (the to-dir argument of move_log_segments), it modifies the log_history_dir in the next newer log of the family to reflect the new directory into which older logs have been moved.

Since move_log_segments only searches for log segments to move in a single directory (the from-dir argument), it will never move all the log segments from that directory to the to-dir. The command ALWAYS leaves at least one segment (the newest segment in the from-dir). It must do this because it doesn't know what directory contains the next newer log in the family. For example, in the diagram above, the command:

```
move_log_segments iolog >udd>sa>old_logs >udd>sa>old_iologs 1day
```

could move iolog.19861111.151709 to the old_iologs directory, but would leave iolog.19861113.074657 because it cannot know that the next earlier log in the family is located in >udd>sa>a>h. After the command above, the diagram would look like:

LOG DIRECTORY	LOG SEGMENT	LOG HISTORY DIR (in log header)
>sc1>as_logs	iolog	>sc1>as_logs
.....	iolog.19861119.063822	>udd>sa>a>h
>udd>sa>a>h	iolog.19861117.223517	>udd>sa>a>h
.....	iolog.19861115.093546	>udd>sa>old_logs
>udd>sa>old_logs	iolog.19861113.074657	>udd>sa>old_iologs
>udd>sa>old_iologs	iolog.19861111.151709	(null string)

NOTES:
The cutoff of 1day is a time in the future. The move_log_segments will then attempt to move all segments older than that time.

move_log_segments will move all log segments except one, the newest log segment in the from-dir.

The log segment that was not moved has had its log_history_dir value set properly.

The final segment in the old_logs directory can be moved with the move command, providing the log_history_dir in the next newer log gets updated to reflect the new location of the moved log.

```
move >udd>sa>old_logs>iolog.19861113.074657 >udd>sa>old_iologs>==
```

The oldest iolog in >udd>sa>a>h is found and the set_log_history_dir command is used as follows:

```
set_log_history_dir iolog.19861115.093546 >udd>sa>old_iologs
```

The print_sys_log command (and associated commands) should now be able to find the logs just processed. The diagram will then look like:

LOG DIRECTORY	LOG SEGMENT	LOG HISTORY DIR (in log header)
>sc1>as_logs	iolog	>sc1>as_logs
.....	iolog.19861119.063822	>udd>sa>a>h
>udd>sa>a>h	iolog.19861117.223517	>udd>sa>a>h
.....	iolog.19861115.093546	>udd>sa>old_iologs
>udd>sa>old_iologs	iolog.19861113.074657	>udd>sa>old_iologs
>udd>sa>old_iologs	iolog.19861111.151709	(null string)

REPORTS

The crank uses the summarize_sys_log command to generate daily reports of messages appearing in the logs.

SECTION 25

SECURITY AUDITING

Multics provides the capability to audit security related events. These events are recorded in either the:

- syserr log
- answering service log
- message coordinator logs, or
- the admin log

Messages are routed to these logs based upon the subsystem that generates them and the audit selection criteria that you specify.

A security audit trail consists of a series of messages inserted into the system logs at the time some security related event occurs. The logs which may have security audit messages are the syserr log, the answering service log, the admin log, and the message coordinator logs. Most of these messages are in a standard format and are easily recognized; a few audit messages are nonstandard. Both types are described in detail in this section.

STANDARD AUDIT MESSAGES

Every standard log message consists of a variable length text string and, additionally, binary data which can be interpreted into a text string and printed using the -interpret control argument to the print_sys_log command (see "Perusal and Analysis of Audit Logs" later in this section). Associated with binary data is a data class. Standard security audit messages contain binary data of class access_audit. (For a complete description of data classes, refer to the print_sys_log command in the *Administrative, Maintenance and Operations Commands* manual). Most of the standard messages have text that begins with the characters "Audit".

Audit messages are considered nonstandard if there is no binary data associated with the message.

BINARY DATA HEADER FORMAT FOR STANDARD AUDIT MESSAGES

The binary data for a standard audit message consists of one or more parts. The first part is always a header as defined by the structure "audit_record_header" found in access_audit_bin_header.incl.pl1. The data expander, invoked when the "-interpret" control argument is supplied to the "print_sys_log" command, displays this data in the following format:

```
{Proxy: <user_id> (ring <level>), PID=<process_id>
Auth: <process_auth>, Min: <min_auth>, Max: <max_auth>}
Subject: <user_id> (ring <level>){, PID=<process_id>}
                          {, Session_UID=<session_uid>}
Auth: <process_auth>, Min: <min_auth>, Max: <max_auth>
<object_type>, operation type: <oper_type>
                          {, detailed operation = <detail>o}
```

where:

The first two lines beginning with "Proxy" are present if the operation was performed by a privileged process on behalf of the subject process. These first two lines reflect information about the privileged proxy process.

<user_id>
is the Person.Project.tag of the user/proxy

<level>
is the validation level at the time of the operation or proxy request

<process_id>
is the user/proxy ID expressed as an octal word

<session_uid>
is a unique identifier used by the answering service for terminal session identification (across processes)

<process_auth>
is the user/proxy authorization expressed in octal as a level and categories: L:CCCCCC

<min_auth>
is the minimum authorization for this user_id

<max_auth>
is the maximum authorization for this user_id

<object_type>
may be one of:
No_Object
File_System_Object
File_System_Attributes
RCP_Object
Administrative_Object
Special_Object
Other_Object

<oper_type> may be one of:
No_Operation
Modify_Access
Modify
Read

<detail>
is a refinement of the operation listed in the text of the message. It is given (in octal) only if non-zero. There are only two areas of the system that use the detailed operation field - RCP and the file system. In the case of RCP the 18 available bits are used as flags and are defined in rcp_opr.incl.pl1. The file system uses the 18 bits as an integer code for the operation. These codes are defined in fs_obj_access_codes.incl.pl1. Both these include files appear in the appendix.

All standard audit messages contain binary data that begins with this information. It may be followed by additional binary data. These usually supply more detailed information about the object involved in the event. Thus, the format is different for different types of objects. These are described later in this section.

AUDIT MESSAGES IN THE SYSERR LOG

The messages in the syserr log include a text string portion and, for standard messages, a detailed binary data portion in addition to the binary header information previously described. The syserr log also contains nonstandard messages which do not contain interpretable binary data.

Text String Portion of Syserr Log Audit Messages

The standard messages in the syserr log can include several categories of information that define the event that has occurred based on the audit selection criteria employed. These categories are:

- system object - the type of object
- operation status - this is either "GRANTED" or "DENIED"
- operation type - the type of operation performed on the object
- operation code - a specific indicator of the operation
- operation mode - any special mode under which the operation was attempted

Listed below are descriptions of the system objects, operation status, operation type, operation code, and optional operation mode that are combined to produce an audit message in the syserr log.

SYSTEM OBJECTS

The following is a list of system objects upon which user related events can be audited:

- file system objects (fsobj)
- file system attributes (fsattr)
- RCP objects (rcp)
- administrative objects (admin)
- special objects (special)
- other system objects (other)

The abbreviated names appearing in parenthesis are used in commands related to security auditing.

File System Objects

File system objects are segments, directories, or links. (Multisegment files and data-management files are not considered as single entities, but as directories and segments). Operations possible on file system objects are "read" or "write" (modification of system object).

File System Attributes

File system attributes are the attributes associated with a file system object. They have been separated from the file system object category for ease in audit selectivity (see "Syserr Log Selectivity" below). These attributes include such things as the various switches and the access control list. Any of the three system operations apply to file system attributes (i.e. "read", "write", or "modify access").

RCP Objects

RCP objects are any of the entities controlled by the Resource Control Package. They include tape and disk drives, consoles, tape and disk volumes, etc. Any of the three system operations apply to RCP objects.

Administrative Objects

Administrative objects are special file system objects. These are typically data bases which are used in the administration (typically registration and accounting) of the system. One example is the Person Name Table (PNT).

Special Objects

Special objects are those which may be manipulated only by highly privileged processes. For example, a process is considered a special object which may be manipulated only by the Initializer.

Other Objects

Other objects are any which do not fall into the above categories such as mailbox entries.

To enable auditing for one or all of the above system objects, see "Syserr Log Audit Selectivity", below.

OPERATION STATUS

An operation status can be either "GRANTED" or "DENIED". This indicates whether the operation performed on the system object being audited was allowed or disallowed, respectively. You have the option to tell the system to create audit records that show only operations that were granted, only operations that were denied, or you can create audit records that include both grants and denies of the various operation types (see below) that can be attempted on a system object.

OPERATION TYPES

There are three operation types performed on system objects that can be audited. They are:

- read
- modify access
- modification of object

"Read" implies that a process attempted to read a system object; "modify access" implies that a process attempted to modify the access of a specific system object; "modification of object" implies that a process attempted to change or modify a specific system object.

You have the option of creating audit records that record any or all of the above operation types performed on a system object. For example, you can elect to audit all successful (GRANTED) attempts to modify a file-system object (fsobj). For detailed information on syserr log audit selectivity, see "Syserr Log Audit Selectivity", below.

OPERATION CODE

An operation code is a unique identifier for the specific operation that was carried out or attempted. This is an entry in the access_operations_ table that is translated into a textual description of the operation.

OPERATION MODES

The following is a list of operation modes that can be performed on system objects and, optionally, recorded in the audit message of the syserr log:

- privileged operation (priv_op)
- administrative operation (admin_op)
- illegal procedure or access violation faults (faults)
- small covert channel operations (small_cc)
- moderate covert channel operations (moderate_cc)
- special operations (special)

The absence of any mode indicator is a normal user operation.

Privileged Operation

Privileged operation is one performed through the use of a privileged gate or under previously set AIM privileges. These operations are typically used for system maintenance.

Administrative Operation

Administrative operation is one performed for normal system operation. This includes registering users, manipulation of resources, etc.

Illegal Procedure and Access Violation Faults

Illegal procedure and access violation faults are situations detected by the hardware which may indicate an attempt to access protected data.

Small and Moderate Covert Channel Operations

Small and moderate covert channels are those which have a potential bandwidth of 1-10 bps and 10-100 bps, respectively.

A covert channel is a mechanism by which a process may signal (and thereby pass information) to another process via indirect means. The important case is where the process sending the signal is at a higher authorization than the process reading (receiving) the signal. This type of operation status may occur in conjunction with a successful or unsuccessful operation.

Signalling mechanisms typically make use of a resource accessible to both the high and low authorization processes. By manipulating such resources in an agreed upon manner, a code may be transmitted.

For example, suppose the high authorization process is to send a "yes" or "no" signal to a low authorization process. An agreement is made where tape_drive #23 will be used to communicate the signal. The low authorization process will attempt to attach tape drive #23. If successful, the result is "no". If unsuccessful because the resource is in use, the result is "yes". Assuming the sender and receiver are the only two users logged in, the results are accurate. Of course, some third party not involved in the covert communication may just happen to have attached tape_drive #23. This means the result was wrong, other processes are introducing noise into the channel. However, given more than one resource, and clever error detection/correction schemes, accurate information may be transmitted.

Clearly, it would be easier for a user at a high authorization to transmit information to the low authorization user outside the system (e.g. telephone conversation). The concern here is of high authorization users leaking information without their knowledge. This situation arises when a "Trojan Horse" program planted by a low authorization user is executed by the high authorization process. Trojan Horse programs are typically disguised as an innocuous looking tool or game. A side effect programmed into them can make use of covert channels to transmit information to their owner. Thus, the need to audit uses of covert channels.

Covert channels are categorized by their bandwidth. That is, how much information can be transmitted in a given time interval. The categories are:

```
trivial   -   <1 bits/second
small     -   1-10 bits/second
moderate  -   10-100 bits/second
large     -   >100 bits/second
```

Trivial covert channels are small enough that the system need take no action against them. Large covert channels are considered security breaches and may not exist in a secure system. Thus, the only applicable categories are the small and moderate covert channels. All known small and moderate covert channels may be audited. They are regarded separately so that the site may chose to audit one, the other, both, or neither (see "Syserr Log Audit Selectivity" below).q

Special Operations

Special operations are security related events which are always audited. These include the reclassification of objects.

STANDARD SYSERR LOG AUDIT MESSAGE TEXT STRING FORMAT

All standard format audit messages in the syserr log are generated by either access_audit_, page_error, or access_audit_log_fault_. The general format of the audit message text string generated by access_audit_ and page_error is:

```
Audit (<module_name>): <GRANTED | DENIED> <operation_desc>
     (<event_flags>) for <proxy/user_id> (<proxy/user_auth>)
     Level=<proxy/user_level>
     {on behalf of <user_id> (<user_auth>)}
     {to {<obj_type>} <obj_name> (<obj_class>)}
     {<extra_text>}. {Code=<error_code_text>}
```

where:

<module_name>
is the name of the module which called access_audit_ or the string "page_error"

<operation_desc>
is the textual description of the operation extracted from the access_operations_ table. The appendix contains a list of all the operation descriptions which may appear in the syserr log.

<event_flags>
are names of any event flags which were on. These include:

special_op
: an operation alway audited to make it highly visible

admin_op
: an administrative operation

priv_op
: an operation performed using a privileged entry or AIM privileges

small_cc
: operation can be used to effect a small bandwidth covert channel (1-10 bps)

moderate_cc
: operation can be used to effect a moderate bandwidth covert channel (10-100 bps)

receiver
: operation which might be used as a covert channel is on the receiver side

<proxy/user_id>
is the Person.Project.tag of the process which performed the operation. This may be a privileged proxy process which performed the operation on behalf of another user

<proxy/user_auth>
is the authorization of the process which performed the operation

<proxy/user_level>
is the validation level of the process which performed the operation

<user_id>
is supplied if the operation was performed by a proxy process. It is the Person.Project.tag of the process which requested the operation

<user_auth>
is the authorization of the process which requested the operation

<obj_type>
for objects supported by ring-0 this will be "segment", "directory" or "link". Other types include "tape_drive", "tape_vol", "disk_drive", etc. as supplied by the ring-1 subsystem.

<obj_name>
is the name or pathname of the object. This may not be included for those operations which do not operate on standard objects.

<obj_class>
is the access class of the object involved.

<extra_text>
is extra information which may be supplied by the caller of access_audit_. It is free-form text which is self-defining.

<error_code_text>
is the result of convert_status_code_ if the code supplied to access_audit_ is non-zero.

The text string format of standard messages generated by access_audit_log_fault_ is:

```
Audit (fim): <fault_name> <fault_subname>
      by <user_id> (<user_auth>) Level=<level>
      at {SLT seg} <proc_segno>|<proc_offset>
                {<proc_name>|<proc_offset>} (<proc_class>)
      referencing {SLT seg} <data_segno>|<data_offset>
                {<data_name>|<data_offset>} (<data_class>)
      {Mode=<modes> Rings=<rings>}
      Inst=<even_inst_word>, <odd_inst_word>
      {(with <n_locks> locks set)}.
```

where:

<fault_name>
may be one of:
illegal procedure fault
access violation fault - ring
access violation fault - mode

<fault_subname>
may be:
(for IPR faults):
no fault - bad call (program/hardware error)
inv fault (never happen)
illegal op code
illegal addr/modifier
illegal slave proc
illegal procedure
non-existant addr
out of bar bounds
bad ipr type - bad call (program/hardware error)

(for ACV faults):
no fault - bad call (program/hardware error)
illegal ring order
not in execute bracket
no execute permit
not in read bracket
no read permit
not in write bracket
no write permit
not a gate
not in call bracket
outward call
bad outward call
inward return
cross-ring transfer
ring alarm
assoc. memory fault
out of segment bounds
bad acv type - bad call (program/hardware error)

<user_id>
is the Person.Project.tag of the process which encountered the fault

<user_auth>
is the authorization of the process which encountered the fault

<user_level>
is the validation level of the process at the time it encountered the fault

<proc_segno>
is the segment number of the procedure which encountered the fault

<proc_offest>
is the offset within the procedure

<proc_name>
is the name or pathname of the procedure

<proc_class>
is the access class of the procedure. May also appear as:

"hardcore segment"
"unable to get entry"
"invalid seg#"
"deleted seg - null ref name"
"deleted seg"
"unable to determine path"

<data_segno>
is the segment number of the data segment being referenced at the time of the fault

<data_offset>
is the offset within the data segment

<data_name>
is the name or pathname of the data segment

<data_class>
is the access class of the data segment. May also appear as the one of the strings given for <proc_class>.

<modes>
are the process's effective modes on the data segment (displayed for ACV faults only)

<rings>
are the ring brackets of the data segment (displayed for ACV faults only)

<even/odd_inst_word>
is the even/odd instruction word pair being executed at the time of the fault

<n_locks>
is the number of ring-0 locks being held by the process at the time of the fault (displayed if non-zero)

The following are sample messages from a syserr log:

```
Log >sl1>syserr_log from 1985-02-26 11:43:18.865117 to 1985-02-26
11:49:55.018252

1985-02-26  Tue  est

11:43:26 1037819 34 Audit (set_privileges): GRANTED modification of
                    system AIM privilege (Special_op Priv_op) for
                    Pandolf.SysMaint.a (7:777777) Level=4 {ring one
                    turned on}.

11:43:37 1037820 44 Audit (dc_find): GRANTED initiation of fs_obj
                    (Priv_op) for Pandolf.SysMaint.a (7:777777) Level=1
                    to segment >udd>m>map>Pandolf.mbx (0:000000).
```

If the content of a syserr log audit message is not detailed enough for your site's purpose, print the log using the -interpret control argument to the print_sys_log command. The -interpret control argument produces a binary representation of the message.

Detailed Binary Data Portion of Syserr Log Audit Messages

In addition to the standard binary data header described early in this section, audit messages in the syserr log can contain a second set of binary data for:

- File system Objects
- PNT entries
- RCP objects
- Message segment entries
- Processes

The format of the binary message for each is described below.

Note that there are also some operations which operate on no object in particular. These may set privileges, audit flags, etc. All pertinent information about the event is included in the text of the syserr message. Of course, the binary header information described earlier in this section is available in these messages.

DETAILED BINARY DATA FORMAT FOR FILE SYSTEM OBJECTS

The binary data following the standard binary header previously described in this section is defined by the structures "audit_ssobj_info" and "audit_link_info" in access_audit_ssobj_info.incl.pl1. The interpretation performed by print_sys_log adheres to the following format:

```
Object: <branch|link> <uid> in <parent_path>,
DTEM is <date_time_str>
Raw mode: <mode> Ring brackets: <rings> Class: <acc_class>
{Ex mode: <ex_mode>  Ex ring brackets: <ex_rings>}
Switches: <branch_switches>
```

where:

<branch|link>
is one of these literal strings. For links, only this first line of the interpreted data is given.

<uid>
is the file system unique ID associated with the entry

<parent_path>
is the pathname of the object's parent directory as derived from the UID path of the object

<date_time_str>
is a date-time formatted string

<modes>
is the user process's effective access mode on the object

<rings>
are the ring brackets on the object

<ex_modes>
are the extended access modes on the object

<ex_rings>
are the extended rings on the object

<acc_class>
is the AIM access class label represented as an octal level and categories: L:CCCCCC

<branch_switches>
is a list of the directory entry switch settings for the object. Each switch is separated by a comma and may be preceeded by a caret (^) to indicate its value is "off". The switches are: dirsw, per_processs, safety, multiple_class, audit, security_oos, entrypt, master_dir. (Note that the "audit" switch has no meaning in MR11)

DETAILED BINARY DATA FORMAT FOR PNT ENTRIES

The binary data following the standard header for PNT entries is defined by the structure "pnt_audit_record" in pnt_audit_record.incl.pl1. The interpretation performed by print_sys_log adheres to the following format:

```
User id = <person_id>, Operation = <operation_type>
                {Changed password} {Changed network password}
New PNT info:
     Alias = <alias>,
     Authorization range = <min_auth>-<max_auth>,
     Audit flags = <process_audit_flags>,
     Flags = <pnt_flags>
{Old PNT info:
     Alias = <alias>,
     Authorization range = <min_auth>-<max_auth>,
     Audit flags = <process_audit_flags>,
     Flags = <pnt_flags>}
```

where:

<person_id>
is the name of the entry involved

<operation_type>
is the operation performed. It may be "add", "delete", "modify", or "unknown".

<alias>
is the alternate string by which a person_id may be used

<min_auth>, <max_auth>
are access authorization values represented as octal level and categories (i.e. L:CCCCCC). The two values represent the range of authorizations at which the person_id may be used.

<process_audit_flags>
is a list of the audit flags applied to a process created for the person_id. See the "new_user" command in *Administration, Maintenance and Operations Commands* manual, Order No. GB64 for a description of these flags.

<pnt_flags>
are literals describing the flags associated with a PNT entry. They are listed separated by commas and may be preceeded by a caret (^) to indicate their value is "off". The possible flags are: password, network_pw, trap, lock, change, must_change, generate, operator, and time_lock.

DETAILED BINARY DATA FORMAT FOR RCP OBJECTS

The binary data following the standard header for RCP objects is defined by the structure "rcp_obj_info" in access_audit_rcp_info.incl.pl1. The interpretation performed by print_sys_log adheres to the following format:

```
Type: <resource_type>{ registry}, Name: <resource_name>,
Owner: <owner_id>,
Access class: <min_auth>-<max_auth>,
Raw mode = <mode>, Ring brackets = <rings>.
Attributes: <attributes>
Flags: <rcp_flags>
```

where:

<resource_type>
is the type of the resource (tape_vol, disk_vol, tape_drive, disk_drive, special, etc.).

<resource_name>
is the name of the resource.

<owner_id>
is the resource owner which may be a Person.Project, "system", "scratch", or "free".

<min_auth>,<max_auth>
are access class values expressed as octal level and categories (i.e. L:CCCCCC). Together they form a range within which the resource may be used.

<mode>
is the process's raw access mode to the resource.

<rings>
are the ring brackets, if defined, for the resource.

<attributes>
are the rcp_attributes associated with the resource. It is derived from the binary attributes by cv_rcp_attributes_$to_string.

<flags>
is a list of flags associated with the resource. The list of literals is separated by commas and each may be preceeded by a caret (^) to indicate its value is "off". The flag are: device, volume, usage_locked, release_locked, awaiting_clear, has_acs_path.

DETAILED BINARY DATA FORMAT FOR MESSAGE SEGMENT ENTRIES

The binary data for message segment entries (i.e. messages) following the standard header is in two parts. The first part is the same as for file system objects which was described earlier in this section. The second part is defined by the structure "audit_mseg_msg_info" in access_audit_mseg_info.incl.pl1. The interpretation of this second part performed by print_sys_log adheres to the following format:

```
MSEG V5 descriptor:
Sender id=<sender_group_id>
Sender level=<sender_validation_level>
Sender pid=<sender_process_id>
Sender authorization=<sender_auth>
Sender max authorization=<sender_max_auth>
Sender audit=<audit_flags>
Message ID=<message_id>
Access class=<message_access_class>
```

where:

<sender_group_id>
is the Person.Project.tag of the process which sent the message

<sender_validation_level>
is the validation level of the sending process

<sender_process_id>
is the process_id of the sending process

<sender_auth>
is the authorization of the sending process expressed as octal level and categories (L:CCCCCC)

<sender_max_auth>
is the maximum authorization of the sending process expressed as octal level and categories (L:CCCCCC)

<audit_flags>
are the per-process audit flags of the sending process expressed as an octal word

<message_id>
is the unique ID of the message

<message_access_class>
is the access_class of the message expressed as octal level and categories (L:CCCCCC)

DETAILED BINARY DATA FORMAT FOR PROCESSES

The binary data for processes consists of just the standard header as described earlier in this section. The only operation on a process logged in the syserr log is an IPC wakeup. The pertinent information about the target process is contained in the text of the message (i.e. its process_id).

Nonstandard Syserr Log Audit Message Format

There are nonstandard audit messages that appear in the syserr log. These messages record the use of some privileged operations available in the system and record:

- changes made to volume registration data
- overflow of the interprocess transmission table
- master directory control subsystem repair
- when normal syserr logging has been disabled

The nonstandard audit message generated when changes are made to volume registration data is:

```
volume_registration_manager_: <operation> by <user_id>.
```

where:

<operation>
is a text string describing the operation which took place

<user_id>
is the group_id of the privileged user who requested the operation

These messages describe changes made to the volume registration data. Some of the changes possible are related to volume access. They are all documented in the system message document distributed with the release (>doc>MR11.0>em.compout). This document is arranged in alphabetical order for easy reference.

The nonstandard audit message generated when overflow of the interprocess transmission table occurs is:

```
hc_ipc_: ITT overflow caused by <user_id>.
```

Messages in the above format record the overflow of the inter-process transmission table. Since this table is used by processes of all access authorizations, it can by used to signal information across AIM boundaries (i.e. a covert channel). However, the bandwidth is low and each instance is logged and printed on the console. Therefore, it is very unlikely to be useful.

The nonstandard audit message generated when the master directory control subsystem has repaired damage to its data base is:

```
mdc_repair_$ENTRY: <message>
```

There are a number of modules that note, on the console, when normal syserr logging has been disabled due to some system failure. They are generated from the following modules and are all documented in the system message document:

- syserr_copy
- syserr_copy_wired_buffer
- syserr_log_man_
- syserr_seg_manager

Syserr Log Audit Selectivity

Syserr log audit selectivity, or determining what actually gets recorded in the syserr log, must be configured on two levels:

- system level
- process level

Each is described in detail below.

SYSTEM AUDIT FLAGS AND THRESHOLDS

Syserr log audit defaults must be set at the system level with the set_system_audit_flags command (see the *Multics Administration, Maintenance and Operations Commands* manual, Order No. GB64). The set_system_audit_flags command sets system parameters that control auditing of user access to system resources. This is further controlled by the per-process audit flags (see below). The set_system_audit_flags command lets you set default auditing of:

- covert channels
- successful access to system resources
- unsuccessful access to system resources

You can display the system audit flags by issuing the display_system_audit_flags command (see the *Administration, Maintenance and Operations Commands* manual, Order No. GB64).

System audit flags in the "enabled" state have an associated audit threshold. The format of this threshold is the same as an access class or authorization. That is, it includes a level any combination of categories. A test access class/authorization is within the threshold if either of the following two statements is true:

```
TEST_LEVEL >= THRESHOLD_LEVEL

(CATEGORIES & THRESHOLD_CATEGORIES) ^= "000000"b3  {logical AND}
```

Note that none of the system audit flags control auditing of operations designated as special, privileged, administrative, or faults. The latter three are controlled only by the individual process audit flags. Operations designated as special are always audited by the system (object reclassifications are presently the only special operations).

Covert Channel Use

For covert channel events in a transmitting process, auditing is performed if you enable the system covert channel audit flag and the process's authorization is within the specified threshold. For a receiving process, auditing is performed if the system covert channel audit flag is enabled, the threshold does not limit auditing in this instance.

Auditing Successful Access to System Resources

Auditing of successful access to system resources lets you specify that all operations on system resources which are granted by the system are to be audited. Only those operations involving resources equal to or above the specified threshold are audited.

Auditing of Unsuccessful Access to System Resources

You can also audit all unsuccessful attempts to access system resources. Only those operations involving resources equal to or above the specified threshold are audited.

PROCESS AUDIT FLAGS

Process audit flags are built from:

- project audit flags stored in the SAT for the project
- person audit flags stored in the PNT entry for the user

Setting Default Process Audit Flags

You can define default audit flags for all new users and new projects on the system by specifying audit flags in >udd>SysAdmin>admin>sys_admin_data with the admin_util command (see the *Multics Administration, Maintenance and Operations Commands* manual. Order No. GB64).

Setting Audit Flags for Projects

For projects, set the audit flags at project creation time with the new_proj command. For existing projects, you can change the audit flags with the edit_proj command. To display the current audit flags for an existing project, use the print_sat command. For a complete description of each of these commands, see the *Multics Administration, Maintenance and Operations Commands* manual, Order No. GB64).

Setting Audit Flags for Users

To set audit flags for users, use the new_user command when registering a user for the first time; for existing users, use the new_user$cga command to change the flags (see the *Multics Administration, Maintenance and Operations Commands* manual, Order No. GB64). You can display the audit flags for a user by issuing the print_pnt command.

Audit criteria for projects and users must be specified in the following format when using the new_proj or new_user commands:

```
<object_type>=<grant_level>/<deny_level>
```

where <object_type> may be "fsobj", "fsattr", "rcp", "admin", "special", or "other" as previously described in "System Objects". The <grant_level> specifies successful access and the <deny_level> specifies unsuccessful access. The <grant_level> and <deny_level can be "N", "MA", "M", or "R" representing NONE, MODIFY_ACCESS, MODIFY, or READ audit levels as described below:

- NONE - No auditing
- MODIFY_ACCESS - Auditing of operations which change the ability of others to access the object in question
- MODIFY - Auditing of operations which change the object itself, or its attributes/properties. This level includes the MODIFY_ACCESS operations
- READ - Auditing of operations which return information about the contents of the object or its attributes/properties. This level includes the MODIFY and MODIFY_ACCESS operations. It allows the maximum amount of auditing.

In addition to the above format for audit criteria specification, the following five additional "binary state" flags (enabled or disabled) can be set:

- priv_op
- admin_op
- fault
- small_cc and moderate_cc

as previously described in "Operation Modes". Specification of these flags is either "^operation_mode" to indicate that the named flag is disabled, or "operation_mode" (without the ^ prefix) to indicate that the named flag is enabled.

The priv_op flag controls auditing of privileged operations performed by the process. This includes such operations as setting AIM privileges. It is recommended that sites interested in auditing should turn this flag on for all processes except perhaps the system daemons.

The admin_op flag controls auditing of administrative operations performed by the process. This includes operations such as registration of new users. It is recommended that sites interested in auditing should turn this flag on for all processes.

The fault flag controls auditing of some types of access violation and illegal procedure faults raised in the process. Repeated occurrence of these faults may indicate a repeated effort to subvert system security (although they may be due to innocent programming errors).

The small_cc and moderate_cc flags control auditing of covert channel activity (contingent on the system covert channel audit flag and threshold). It is recommended to sites interested in covert channel auditing to turn these flags on in all processes except those of trusted system_low users and the system daemons.

The following example is a printed representation of the audit flag specification for a process:

```
fsobj=N/M,fsattr=MA/MA,rcp=M/R,admin=R/R,special=R/R,...
   ...other=MA/R,admin_op,priv_op,fault,small_cc,moderate_cc
```

Process audit flags are constructed at process creation time from the flags stored in both the PNT entry for the user and the SAT entry for the project. For each object audit level, the greater value is used. For each audit flag, the logical "or" of the two is used.

SAT and PNT audit flags may be changed using the edit_proj and new_user$cga commands respectively. Only those elements requiring change need be entered. Thus, if you want to change the rcp object audit levels you need only enter "rcp=X/Y", the other values will be left intact. (Note: after using edit_proj on >udd>SysAdmin>admin>smf.cur.sat you must use "install smf.cur.sat -auth | -all" to make the change effective). After changing the audit flags in a PNT or SAT entry the affected users must log out and log in again. Simple "new_proc" will not pick up the new values for the audit flags.

AUDIT MESSAGES IN THE ANSWERING SERVICE LOG

Six types of system events are recorded in the answering service log:

- Identification and Authentication (login and logout)
- process manipulation
- attempts to attach communications channels
- dial services
- daemon manipulation
- console I/O

Each of the types of information recorded in the answering service log is described in detail below.

Identification and Authentication (I&A)

When a user logs in to the system, the user's person_id is identified (Identification); when the password is entered, the system authenticates (Authentication) that the user is actually the person_id entered. I&A messages perform three functions:

- security auditing
- a journal of all system logins and logouts
- notification of user logins for operators

Note that since the answering service log contains information about aborted login attempts, mistyped user passwords may be identifiable in the log. It is important that you safeguard the answering log from unauthorized access.

The format of I&A messages recorded in the answering service log is:

```
LOGIN {DENIED}  <user_id>  <process_type>   <channel_id>
                        {[<absin_entry>]} {(<added_info>)}
LOGOUT          <user_id>  <process_type>   <channel_id>
                           {<cost_info>}   (<added_info>)
```

where:

<user_id>
is the Person.Project of the authenticated user at login, or Person.Project.Tag of the user at logout. An anonymous Person identifier is preceded by an asterisk ().

<process_type>
is one of the following types of authentication:

```
int     an interactive process
Q n     an absentee process (where n is the background
        queue number or FG for a foreground process)
dmn     a daemon process
opr     a message coordinator operator login
```

<channel_id>
is the FNP channel name for an interactive process, the absentee slot number for an absentee process, the message coordinator stream for a daemon process, or the message coordinator console name for operator users.

<absin_entry>
is the final entryname of the absin file associated with an absentee login.

<added_info>
is a string giving circumstances for the login authentication or the logout. For interactive, absentee, and daemon processes, this will be a mnemonic string generated by the software. This may be the login word used, the disconnected process control argument given to login, the reason for the logout, etc. Some examples are: login, enter, enterp, create, connect, new_proc, logo, autologout, lhbr, bad_pers, bad_pass, term. For operator users, it will be "sign_on" or "sign_off".

<cost_info>
is the VCPU time (in minutes and seconds) and dollar cost for the session.

I&A messages are routed to the operator console as well as the AS log. The following are sample I&A audit messages recorded in the answering service log.

```
LOGIN        RRitter.TSDC int c.h000.002 (create)
LOGIN        Wallman.Multics int b.h010.001 (connect loop)
LOGIN DENIED EMarch.BETA int c.h000.001 (bad_pass)
LOGIN        TR_Admin.TR Q FG abs2 [update_user_regs] (create)
LOGIN DENIED Network_Server.Daemon Q 4 abs2 (umxbg)
LOGOUT       Hirneisen.SysAdmin.a int c.h016.001   0:38 $1.05 (lobr)
LOGOUT       Volume_Dumper.Daemon.z dmn vcons  15:25 $18.77 (logo)
```

The interpreted binary data for I&A Audit messages consists of the standard header information described earlier in this section. LOGOUT messages have no additional binary data. LOGIN messages have extra binary data that is interpreted by print_sys_log as follows:

```
Process type = <process_type>,
Min ring = <min_ring>, Max ring = <max_ring>,
Attributes = <process_attributes>,
Audit flags = <process_audit_flags>,
Channel = <channel_id>,
Terminal type = <term_type>,
Answerback = <answerback>,
{Absentee input path = <absin_path>}
```

where:

<process_type>
may be interactive, absentee, daemon, or operator

<min_ring>
is the minimum ring number at which the process is allowed to be authenticated

<max_ring>
is the maximum ring number at which the process is allowed to be authenticated

<process_attributes>
are the attributes for the user combined from the PNT and SAT entries. Only those which are "on" are listed. A complete list can be found in the description of the "new_user" command.

<process_audit_flags>
are the per-process audit flags for the user combined from the PNT and SAT entries. A description of the flags can be found in the description of the "new_user" command.

<channel_id>
is the name of the login communications channel, the abs user slot number, the message coordinator stream, or the message coordinator console channel name as appropriate for the type of process.

<term_type>
is the type of terminal

<answerback>
is the terminal's answerback

<absin_path>
is the pathname of the absin file associated with an absentee process.

Process Manipulation

The answering service log records events related to user process manipulation. The format of process manipulation audit messages is:

```
<key> <user_id> <channel_id> <process_id> {<reason>}
```

where:

<key>
may be one of: CREATE, DISCONNECT, CONNECT, DESTROY, or CONNECT DENIED.

<user_id>
is the Person.Project.Tag associated with the process

<channel_id> .
is the communications channel name, the absentee slot name, or the daemon stream name as appropriate for the type of user. Note that there are no process manipulations for operator users since there are just authenticated, not given a process.

<process_id>
is the octal process unique identifier.

<reason>
is a mnemonic generated by the system describing the reason for the process creation, destruction, or disconnect. Some examples are: new_proc, login, bump, destroy, logo, hangup, hngp, cpg, init, ucs, term, lhbr, logi, etc. For "CONNECT DENIED" messages this field will be the communications channel authorization and the authorization of the disconnected process.

The interpreted binary data for process manipulation audit messages contains only the header described under "Binary Data Header Format for Standard Audit Messages". There is no additional data specific to process manipulation.

The following are sample user process manipulation audit messages recorded in the answering service log.

```
13:04:53  1123341  O CREATE      Swenson.SysMaint.a a.h028 006600751323 (login)
13:18:11  1123365  O DISCONNECT  EJSharpe.MULTICS.a a.h102 006300751314 (hangup)
13:24:00  1123383  O DESTROY     IO.SysDaemon.z prta 005400021626 (logo)
13:34:33  1123406  O CONNECT     EJSharpe.Multics.a a.h103 006000751314
```

Absentee Process Manipulation

Requests for absentee login and cancellation are recorded in the answering service log. The format of these messages is:

```
ABS LOGIN requested by <user_id> Level=<ring>
ABS CANCEL {DENIED} <user_id> <channel_id> <process_id> (car by <canceller_user_id>)
```

where:

<user_id>
is the Person.Project.Tag associated with the requesting process (for login requests) or the absentee process (for cancellations).

<channel_id>
is the absentee slot name.

<process_id>
is the octal process unique identifier.

<canceller_user_id>
is the Person.Project.Tag associated with the process requesting absentee cancellation.

No binary data is associated with absentee login requests. The interpreted binary data for absentee cancellation audit messages contains only the header described under "Binary Data Header Format For Standard Audit Messages".

The following are some sample absentee process manipulation audit messages recorded in the answering service log.

```
fnt small_typ

11:55:05 1037370  0 ABS LOGIN          requested by Utility.SysDaemon.m Level=4
11:12:38 1041887  0 ABS CANCEL          Lippard.Multics.m abs1 021100131205
```

Communications Channel

The answering service records information relating to communications channels. The types of communications channels events that are recorded are:

- dial-out requests
- slave attachments
- dial-in requests
- test and diagnostic (T&D) attachments

AUDIT MESSAGE FORMAT FOR COMMUNICATIONS CHANNEL ATTACHMENT

The text portion of audit messages associated with communications channel attachment are of the general form:

```
<to_key> {DENIED} {<incoming_user_id>} channel <channel_id>
      to <user_id> <process_id> {<service_info>} {(<reason>)}
<from_key> {DENIED} channel <channel_id>
      from <user_id> <process_id> {<service_info>} {(<reason>)}
```

where:

<to_key>
may be one of: ATTACH, DIALIN, or DIAL SYSTEM.

<from_key>
may be either: DETACH or DIALOUT.

<incoming_user_id>
is the Person.Project given in the -user control argument of the dial preaccess command for DIALIN and DIAL SYSTEM operations.

<channel_id>
is the name of the communications channel.

<user_id>
is the Person.Project.tag of the user requesting the ATTACH, DETACH, or DIALOUT operation, or of the target user process of a dial preaccess command for a DIALIN or DIAL SYSTEM operation.

<service_type>
is the channel service type (dial_out or slave) for an ATTACH or DETACH operation, the destination for a DIALOUT operation, or the dial identifier for a DIAL IN operation.

<added_info>
is one of the strings listed in the appendix which describe additional circumstances about the event

The interpreted binary data for communications channel audit messages bears the standard header as previously described. Following this header is data formatted as follows:

```
Channel name = <channel_id>,
{Current access class = <channel_auth>,}
{Access Class Range = <channel_auth_range>,}
{Current service type = <service_type>,}
{Service type = <service_type>,}
{Terminal type = <terminal_type>,}
Userid = <user_id>
```

where:

<channel_id>
is the name of the communications channel.

<channel_auth>
is the authorization at which the channel will be used.

<channel_auth_range>
is the range of authorizations within which the channel may be used.

<service_type>
is the service type of the channel (login, ftp, mc, slave, dial, dialout, inactive, mpx, or t&d).

<terminal_type>
is the type of the "terminal" on the channel

<user_id>
is the Person.Project of the process which has the channel attached, or the Person.Project of an authenticated user dialing into a process.

The following are examples of communications channel attachment audit messages recorded in the answering service log:

```
DIALOUT         channel f.h024.d02 from GDixon.SysMaint.a 016200115221 Destination=*
ATTACH          channel f.h024.d02 to GDixon.SysMaint.a 016200115221 Service=
\dial_out
DETACH          channel f.h024.d02 from GDixon.SysMaint.a 016200115221 Service=
\dial_out (hangup)
DIALOUT         channel b.h022 from Lippard.Multics.a 016600115447 Destination=
\9 555-7316
DIALOUT DENIED channel b.h022 from Lippard.Multics.a 016600115447
        (Unable to complete connection to external device, error in dialing out)
DETACH DENIED  channel acu, from Lippard.Multics.a 016600115447 Service=
\terminate dial out
        (Resource not known to the system, Channel acu, does not exist.)
DIALIN          channel b.h004.001 to Network_Server.Daemon.z 005400114731 Dial
ID=
\smtp
ATTACH          channel b.h004.001 to Network_Server.Daemon.z 005400114731
Service=
\dial_in
DETACH          channel b.h004.001 from Network_Server.Daemon.z 005400114731
Service=
\dial (hangup)
```

DIAL SERVICE AUDIT MESSAGE FORMAT

The text portion of audit messages associated with dial-id services are of the general form:

```
Audit: {GRANTED|DENIED} dialid service {start|stop}
     for <user_id> (<dial_qualifier>) <added_info>
```

where:

<user_id>
is the Person.Project.tag requesting the dial service

<dial_qualifier>
is the dial ID

<added_info>
is one of the strings listed in the appendix.

The interpreted binary data for these messages consists of the standard header followed by data in the following format:

```
Dial qualifier = <dial_id>,
Dial server ring = <server_ring>,
Flags = <dial_flags>
```

where:

<dial_id>
is the name associated with the dial service

<server_ring>
is the validation level of the server process

<dial_flags>"
may be omitted, "registered" or "privileged"

The following are sample dial service audit messages recorded in the answering service log.

```
08:40:00  1127207   0 Audit:  GRANTED dialid service for Network_Server.
                              Daemon.z (smpt) (registered privileged)
08:40:40  1127212   0 Audit:  GRANTED dialid service start for Internet.Daemon.z
                              (internet) (registered privileged)
09:32:33  1127596   0 Audit:  GRANTED dialid service start for IO.SysDaemon.z
                              (cisl_diablo) (registered)
09:44:13  1127851   0 Audit:  GRANTED dialid service stop for IO.SysDaemon.z
                              (cisl_diablo) dialid released (keeping one
                              dialed console)
```

Nonstandard Audit Messages - Daemon Manipulation

Although the text portion of the messages appears to be of standard format, there is no binary data. The general format of these messages follows below. (Note that messages appear in pairs for the GRANTED case).

```
Audit: GRANTED <operation_desc>
     for <user_id> Level=<level> to <stream>
asr_daemon_command_server_: <user_id>: <key> <stream> {<command_line>}

(input on <channel_id>) <key> <stream> {<command_line>}
Audit: GRANTED <operation_desc>
     for <user_id> Level=<level> to <stream>

Audit: DENIED <operation_desc>
     for <user_id> Level=<level> to <stream>
```

where:

<operation_desc>
is the type of operation selected from:
sending a reply to a daemon
sending a quit to a daemon
logging in a daemon
logging out a daemon
sending a new_proc to a daemon
logging in a a daemon

<user_id>
is the Person.Project.tag of the process which requested the operation

<level>
is the validation level of the requesting process

<stream>
is the message coordinator stream with which the daemon process is associated

<key>
is REPLY, QUIT, LOGIN, LOGOUT, or NEW_PROC

<command_line>
are further arguments given to the reply or login daemon commands

Nonstandard Audit Messages - Command Input

The message formats below describe audit messages that log operator console and privileged input to the daemons and answering service. These messages are nonstandard and thus contain no binary data. Their formats are as follows:

```
(input on <mc_channel_id>) <command_line>
```

This message logs input (<command_line>) from a message coordinator terminal (<mc_channel_id>).

```
sc_admin_command_: <user_id>: <command_line>
```

This message logs input (<command_line>) entered via the send_admin_command facility by the user <user_id>.

```
sc_admin_command_: Denied send_admin_command for <user_id>
```

This message logs denial of send_admin_command usage for the user <user_id> whose validation level at the time of the request was <level>.

Nonstandard Audit Messages - Process Termination Monitor

Requests to set the process_termination_monitor are recorded in the answering service log. They have one of the following formats:

```
dpg_:  GRANTED process_termination_monitor request by <user_id>.
dpg_:  DENIED process_termination_monitor request by <user_id>.
```

where:

<user_id>
is the Person.Project.Tag associated with the requesting process.

No binary data is associated with the process termination monitor request.

The following is a sample audit message recorded in the answering service log.

```
22:33:53 1245027  O dpg_:  GRANTED process_termination_monitor request
                    by Data_Management.Daemon.z.
22:43:37 1372351  O dpg_:  DENIED process_termination_monitor request
                    by Dickson.Multics.a.
```

Nonstandard Audit Messages - Comm Channel Info

Requests to get com_channel_info are recorded in the answering service log. They have one of the following formats:

```
as_com_chn_info_srvr_:  GRANTED com_channel_info request for <user_id>
       on channel <channel_id>.
as_com_chn_info_srvr_:  DENIED com_channel_info request for <user_id>
       on channel <channel_id>.
```

where:

<user_id>
is the Person.Project.Tag associated with the requesting process.

<channel_id>
is the channel identifier for which info has been requested.

No binary data is associated with the com_channel_info request.

The following are some sample audit messages recorded in the answering service log.

```
13:26:53 1145865  0 as_com_chn_info_srvr_:  GRANTED com_channel_info request
                    for Dickson.Multics.a on channel b.h021.
16:05:31 1149863  0 as_com_chn_info_srvr_:  GRANTED com_channel_info request
                    for SA1.SysAdmin.a on channel f.h129.
```

Nonstandard Audit Messages - Note PNT Change

Requests for notifications of PNT changes are recorded in the answering service log. They have one of the following formats:

```
as_request_note_pnt_change_: GRANTED NOTE_PNT_CHANGE request from <user_id>.
as_request_note_pnt_change_: Rejected NOTE_PNT_CHANGE request from
       <user_id>.   Validation level (<ring>) not ring-1.
```

where:

<user_id>
is the Person.Project.Tag associated with the requesting process.

<ring>
is the validation ring level of the requesting process when it is not ring-1.

No binary data is associated with the note_pnt_change request.

The following are some sample audit messages recorded in the answering service log.

```
12:55:57 1249766  0 as_request_note_pnt_change_:  GRANTED NOTE_PNT_CHANGE
                    request from SA1.SysAdmin.a.
09:16:34 1546252  0 as_request_note_pnt_change_:  Rejected NOTE_PNT_CHANGE
                    request from Dickson.Multics.a.  Validation level (4)
                    not ring-1.
```

Nonstandard Audit Messages - Miscellaneous

Several nonstandard miscellaneous security audit messages can appear in the answering service log. They are mostly privilged operations. For instance, all table installations performed via answering service request are audited with nonstandard messages. The formats of these messages can take the following forms:

```
up_sysctl_: <install_desc>
up_sat_: <install_desc>
up_cdt_: <install_desc>
up_pdt_: <install_desc>
up_mgt_: <install_desc>
```

where:

<install_desc>
is a description of the table installed and who installed it. Of major importance are the SAT, the PDTS, and the CDT as all these contain security relevant information.

```
DIALIN {DENIED} <user_id1> <channel_id>
                to <dial_id> <user_id2> {<reason>}
```

where:

<channel_id>
is the name of the communication channel

<dial_id>
is the dial server identifier

<user_id1>
is the Person.Project to which initiated the dialin. This appears when the channel requires PersonID and password authentication for slave dials (i.e. has "check_acs: slave_dial" in the CMF).

<user_id2>
is the Person.Project which is serving the dial-id.

<reason>
is the reason for denial (e.g. bad_pass, etc).

Note that this message will usually be followed by a standard "Audit: Attached channel..." audit message when the target process has accepted the dial-in.

```
dpg_: <user_id> (<process_id>) set/replaced
                          process termination monitor
dpg_: removed <user_id> (<process_id>)
                          process termination monitor
```

These messages note the use of the privileged process termination monitor facility. This facility causes wakeups to be sent to <user_id> whenever some other process is destroyed. This is used, for instance, by the data management daemon to cleanup services started by that process.

```
lv_request_:attached/detached lv <lv_name> to/from <user_id>
```

where:

<lv_name>
is the name of the logical volume

<user_id>
is the attaching process

These messages note attachment to or detachment from logical disk volumes. They are applicable from a security viewpoint only in respect to private volumes. Note that attempts to access volumes is audited also in the syserr log.

There are other sources of potentially pertinent messages. Although they are not strictly audits of object access, they are security related. These are all documented in the system message document and are produced by modules named:

```
lg_ctl_
dial_ctl_
load_ctl_
act_ctl_
dialup_
```

The system security administrator should review the document for messages generated by these modules. (The document is arranged alphabetically by module name.)

Answering Service Log Audit Selectivity

There is no audit selectivity, or selection of auditable events, for the answering service log. The answering service log unconditionally records all answering service events, as described above.

AUDIT MESSAGES IN THE MESSAGE COORDINATOR LOGS

The privileged system daemon processes that perform IO, remote driver, and file transfer functions produce output describing each request they process. This includes printer daemons, local card punch daemons, remote station daemons, and imft daemons. You can direct their output, via the message

coordinator, to one or more logs maintained in >sc1>as_logs. It is the responsibility of site personnel to assure that the proper message coordinator definitions and routings are set up at system initialization time (within the system_start_up.ec). The formats of these described below. Note that these messages are non-standard Audit Messages and contain no interpretable binary data.

The format of messages produced by the I/O Coordinator is:

```
<stream> New driver for device <device_name>
     request type <request_name> (series= <init_request_sequence>)

<stream> Driver logout for device <device_name>
```

The format of messages produced by local and remote I/O drivers is:

```
<stream> Request <request_sequence> <type> <queue>:
     <path> from <user_id> (for <heading> at <destination>
```

The format of messages produced by IMFT drivers is:

```
<stream> Request <request_sequence> <request_name> <imft_type>
     <queue>: <transmit/receive> segment <path1> <from/to>
     <user_id> <as/originally> <path2>

<stream> Using channel <channel_name> for input/output, and
     <access_class_range> as the allowable range of access class
     for data transmission.
```

The format of messages produced by driver request denial is:

```
<stream> *Request <request_sequence>:
     <interpreted_error_code> <added_info>
```

where:

<stream>
is the stream over which the message coordinator communicates with the daemon process. Although these may be any letters some conventions are used:

prtX - printers where X is the device tag
punX - punches where X is the device tag
cord - the IO coordinator
XXXfto - IMFT outbound - XXX is remote site acronym
XXXfti - IMFT inbound - XXX is remote site acronym

<request_sequence>
is a sequence number for the request. It is unique per system bootload. The coordinator "New driver" message indicates the starting sequence number for that driver.

<type>
may be "printer", "punch", or "spool"

<queue>
is in the form: "qN" where N is the request queue number

<path>
is the pathname of the segment for the request

<user_id>
is the Person.Project who submitted the request

<heading>
is the user-settable heading for the output

<destination>
is the user-settable destination for the output

<device_name>
is the name of the device. In the case of remote devices it will be in the form <station>.<minor_device>. For driver logout messages it will be simply <station>.

<request_name>
is the name of request queues to be serviced by the driver

<init_request_sequence>
is the request_sequence number the first request to be serviced by this driver, it is simply incremented for each request processed.

<imft_type>
is "output" or "input"

<path1>
is the source segment/subtree pathname

<path2>
is the target segment/subtree pathname

<channel_name>
is the name of the communications channel to be used for data transmission

<access_class_range>
is in the form L:CCCCCC-L:CCCCCC which delimits the low and high values of access class for files being transferred

<interpreted_error_code>
is the text derived from a non-zero system status code received when access to a file is attempted. The two most important values are:

Incorrect access on entry.
Incorrect access to directory containing entry.
Improper access class/authorization to perform operation.

These indicate the user submitted a request for a file to which his/her process lacked appropriate access. Processing of the request is aborted.

<added_info>
is information describing in better detail what prohibited processing of the request. It may state that the user needs "r" access to the file, or that the file's access class was outside the range allowed for the communications channel.

Message Coordinator Log Audit Selectivity

There is no audit selectivity, or selection of auditable events, allowed for the message coordinator log. The message coordinator log unconditionally records all daemon process output for which log streams are defined.

AUDIT MESSAGES IN THE ADMIN LOG

Two basic categories of information are entered into the admin log:

- all I/O produced while in admin mode
- all I/O produced via the send_admin_command (sac) command

Operators can enter admin mode from a specially designated operator console by typing "admin" and entering a password; administrative commands can then be entered. All of the activity that occurs while in admin mode is recorded in the admin log.

Sufficiently privileged users can send commands to the Initializer process to execute administrative commands via the send_admin_command (sac) command. All input and output produced by the sac command is recorded in the admin log.

The admin log should be reviewed on a daily basis by the system security administrator to ensure that no unauthorized use of admin mode or the sac command has been attempted.

The format of the admin log message that records command input from the message coordinator terminal is:

```
(input on <mc_channel_id>) <command_line>
```

When in admin mode, the format is:

```
input:  <command_line>
```

The following is an example of an admin log entry for I/O produced while in admin mode.

```
(input on a.h004): admin
input: pwd
>system_control_1
r ...
input: ame
   .
   .
   .
```

The log entry shows that admin mode was entered, the user issued the print_working_directory command, and the output from the command was >system_control_1. The user then quit admin mode with the ame command.

The format of the admin log messages that record input entered via the send_admin_command command are:

```
sc_admin_command_: <user_id>: <command_line>
sc_admin_command_: <completed command {with errors}>
```

This pair of messages records the command line (<command line>) submitted by the user (<user_id>) via the send_admin_command facility. Any output generated by execution of the command line appears between these two messages. (Denied command lines are logged in the answering service log as "sc_admin_command_: Denied ..." messages.)

The following is an example of an admin log entry for I/O produced via the send_admin_command command.

```
sc_admin_command_: Margulies.SysAdmin.a: move_dir_quota >ddd +1000 >ddd>idd
+1000

sc_admin_command_: Completed command
```

The following message appears whenever an operator user <opr_user_id> successfully signs on for use of a message coordinator terminal <mc_channel_id>. (Denied sign-ons are logged as "LOGIN DENIED..." messages in the AS log).

```
sign_on: <opr_user_id> signed on as operator on <mc_channel_id>
```

The message:

```
<output_text>
```

is any message not preceded by "(input on", "sc_admin_command_", or "sign_on". It is a recording of the output generated by submitted commands. The output for a command immediately follows the corresponding "(input" or falls between an "sc_admin_command" message pair.

Admin Log Audit Selectivity

There is no audit selectivity, or selection of auditable events, for the admin log. The admin log unconditionally records all I/O produced while in admin mode, and all I/O produced as a result of send_admin_command commands.

PERUSAL AND ANALYSIS OF AUDIT LOGS

The auditing of system events is only the first step in creating a secure system. Once the audit trails have been selected and put in place, systematic perusal and analysis of the logs is necessary to note any potential security problems. Two tools are available that give you complete flexibility to view data

from the various logs. They are the summarize_sys_log command and the print_sys_log command (see the *Administration, Maintenance and Operations Commands* manual, Order No. GB64). These commands permit you to selectively view log data to:

- note unusual system activities
- detect excessive audit messages for specific users
- note denied access to files in restricted hierarchies
- detect operator sign_on denials

Before attempting to view the system logs, you should be familiar with the text of audit messages. As mentioned earlier, the audit message text is derived from the access_operations_ table. To view the text of the various audit messages that can be displayed, type:

```
pr >ldd>h>s>bound_library_2_.s::access_operations_.alm
```

You should also be familiar with the use of the summarize_sys_log command and the print_sys_log command.

INITIALLY SETTING AUDIT FLAGS

To "fine tune" the audit log perusal process, it is recommended that you initially set your system audit flags to the values specified below. You can subsequently remove/add audit trails after you have become familiar with the process. Then examine logs and change flags as required.

It is recommended that you initially set system audit flags and thresholds as shown below.

At the system level:

- covert channels - off
- successful access to system objects - system low
- unsuccessful access to system objects - system low

Do this by using the following commands:

```
set_system_audit_flags -no_covert_channel
set_system_audit_flags -successful_access system_low
set_system_audit_flags -unsuccessful_access system_low
```

At the process level (projects and users), audit flags should be set as follows:

```
fsobj=N/R,fsattr=MA/R,rcp=R/R,special=R/R,admin=R/R,other=R/R,priv_op,
admin_op,faults,^small_cc,^moderate_cc
```

As previously mentioned, set the audit flags for projects at project creation time with the new_proj command; for users, set the audit flags with the new_user command.

AUDITING AND SYSTEM PERFORMANCE

System performance can be affected by the amount of auditing that is being performed. System performance can be degraded if large numbers of extraneous events are specified for auditing that are not

crucial to system security. It is important that your site determine those events which are crucial to system security and audit only those related events. It is recommended that you do not not specify the following auditing sequence simultaneously because it may adversely affect system performance:

```
fsobj=R/N
fsattr=R/N
successful_access=system_low
```

Simultaneous specification of all of the above events can cause the audit log to become cluttered with extraneous information that is not essential to system security and degrade system performance.

SECTION 26

MISCELLANEOUS SECURITY TASKS

Miscellaneous security tasks include:

- Maintaining the physical security of the site
- Proper assignment of Project_ids
- Proper management of administrative passwords and user passwords
- Managing rings
- Logical Volume Management
- Setting Access to GCOS Simulator Segments
- Operator identification and authentication

Additional security tasks might include setting access to the GCOS simulator segments and reviewing software changes in new system releases to check for security breaches.

MAINTAINING PHYSICAL SECURITY

Physical security entails keeping the equipment in the machine room secure. A site is no more secure than the machine room. All of the software security features built in to Multics cannot protect against unauthorized access to disk packs and system tapes.

ASSIGNING PROJECT_IDS

At project creation time, you specify the following specific security attributes for the project and the users associated with the project:

- Minimum and maximum login authorizations for users
- Audit flags

There is a login authorization for the project, the person associated with the project, and a login authorization for person in the PNT which is independent of the project. The authorization range for the project is specified in the SAT, the authorization range for the user associated with the project is specified in the PDT, and the independent login authorization range for the user is specified in the PNT. The user is allowed to login only within the constraints of these values. The narrowest authorization of the project's authorization and the user's authorization is the login authorization of the user, and if applicable the AIM range of the communication channel the user is logged in over.

For audit flags there are project audit flags and user audit flags. Audit flags that are turned on in either place cause audit trails of those specific flags for the project and the user.

MANAGING PASSWORDS

Users can be required to change their passwords on a periodic basis. This procedure is recommended for sites that are security conscious. Also, passwords should not be assigned to daemon processes.

Using the ed_installation_parms command, you can manage the use of passwords with the following statements:

- password_change_interval - specifies the amount of time (in days) allowed before users are required to change their passwords.
- password_expiration_interval - specifies the amount of inactive time (in days) allowed for a password before it becomes invalid.
- password_min_length - specifies the minimum allowable length for a password. The recommended value is six characters.
- password_gpw_length - specifies the number of characters used in a system-generated password. The default is six characters.

For more detailed information on these commands, see the Administration, Maintenance and Operations Commands manual.

Locating the User of a Password Being Used Improperly

If a password is being used improperly from a *hardwired* terminal, you can check the channel name in the log file to determine the location of the terminal where the attempted security breach occurred.

If the channel is not hardwired, it can be a network channel or a dial-up channel using ordinary telephone lines; in this case, detecting the terminal being used to attempt the security breach is more difficult. The phone call must be traced.

Changing the Admin Mode Password

Operators who use the Admin Mode password can do more damage to the system than ordinary users since the operator usually has access to the Initializer process and armed with the Admin Mode password could breach security features. The Admin Mode password is usually kept in an envelope next to the operator terminal. If this password has to be used by the operator for any reason, it should be changed and a new password placed in the envelope. To change the password, use the set_special_password command.

MANAGING RINGS

Normally, users and special processes are logged in to the various rings as shown in the table below.

Table 26-1. Ring Management

Project	Min. Ring	Max. Ring	Default
SysDaemon	1	5	1
SysAdmin	1	5	4
SysMaint	1	5	4
All users	4	5	4

In general, the only reason to log in to a ring lower than ring 4 is to fix, delete, retrieve, or or otherwise manipulate an inner ring object. For example, if you want to volume retrieve a mailbox, you are required to log in to ring 1. It should never be true that a user's log in ring is lower than ring 4 for any application. It is sometimes necessary to have a group of users log in to ring 3 to allow maintenance of objects in that ring.

SECURE LOGICAL VOLUME MANAGEMENT

The administration of logical volumes and master directories must be handled by a trusted system administrator. The administrator entrusted with the management of logical volumes and master directories, must follow the following guidelines:

- Only users registered on SysAdmin (or some other suitable project) can perform volume management or master directory operations.
- The "owner" of each logical volume registered on the system must be a selected administrator registered on the SysAdmin project or Initializer.SysDaemon.
- Only SysAdmin and SysDaemon projects should have "sma" access to the directory ">lv". All others must have "s" access to this directory.
- If an ACS path is specified for the logical volume it must be the path of an ACS segment residing in >lv. The ACL for this segment should allow executive ("E") access only to *.SysAdmin.* and Initializer.SysDaemon.z. The rest of the users who are to use the volume are to be given read/write ("RW") access.

- For each volume there should be only one defined quota account. Its name should be *.SysAdmin.*. This means that only users registered on the SysAdmin account may create/delete master directories and manipulate their quota.

- The master directory owner should never be changed to a non-SysAdmin user ID.

The commands associated with logical volume and master directory manipulation are:

```
add_volume_registration (avr)
list_volume_registration (lvr)
change_volume_registration (cvr)
delete_volume_registration (dvr)
init_vol            /* operator command */
reregister          /* operator command */
create_dir (cd)     /* with the -lv control arg */
list_mdir (lmd)
register_mdir
set_mdir_owner (smdo)
set_mdir_quota (smdq)
set_mdir_account (smda)
set_volume_quota (svq)
delete_volume_quota (dlvq)
check_mdcs
```

SETTING ACCESS TO GCOS SIMULATOR SEGMENTS

GCOS simulator segments are located in >unb. To use the GCOS simulators, users require s (status) access to >unb. To use the GCOS simulator, users require re (read execute) access to the following segments:

- bound_gcos_
- gcos_system_software_
- gcos_second_software_
- gcos_library subroutines_
- gcos_altlib_subroutines_

Note that gcos_system_software_ and gcos_library_subroutines_ are multisegment files; and gcos_second_software_ or gcos_altlib_subroutines_ may be multisegment files if the GCOS software was site installed.

For users to use the gcos_tss (gtss) simulator, grant them r (read) and e (execute) access to the following:

- bound_gcos_tss_
- gtss_starL_
- gtss_Lstar_
- gtss_fast_library_

Note that gtss_starL_ and gtss_fast_library_ are multisegment files; gtss_starL_ may be a multisegment file if the software was site installed.

Also required to use the gtss simulator is rw (read write) access to the following:

- GTSS.MCFC.FILES
- GTSS.MCFC.NAMES
- GTSS.MCFC.CALLERS_1
- GTSS.MCFC.CALLERS_2
- GTSS.MCFC.callers_3
- GTSS.MCFC.CALLERS_4

These segments are used for concurrency control. Although their ring brackets are set to 5,5,5, rw access to them is required. Copies of these files (without w access) should be kept in case the files in >unb have to be recreated. Since these files require write access, you may elect to set access only to those person_ids and project_ids that specifically request it.

Access to the following GCOS utilities in >unb should be set to re (read execute) access for all GCOS and GTSS users:

- bound_gcos_fms_
- bound_gcos_utiltities_
- bound_gcos_tools_

Normally, only system administrators should have access to bound_gcos_tools_

To use the GCOS daemon, users require re (read execute) access to the following:

- bound_gcos_daemon_
- bound_gcos_daemon_tls_
- bound_gcos_user_
- gcos_daemon_stat_
- gcos_daemon_stat2_
- gcos_abs_control_

REVIEWING SOFTWARE CHANGES IN NEW SOFTWARE RELEASES

At some sites, security is of sufficient importance to require a careful review of the software changes in each system release. The purpose of such a review is twofold: to verify, to the satisfaction of site administrators, that the changes do not invalidate the access control mechanisms; and to verify that no mistakes, either accidental or malicious, or tape errors are received in a release. Of particular importance are changes to the ring 0 and ring 1 programs that implement the internal access controls.

To make such an incremental review possible, a starting point must first be established. Reviewers should satisfy themselves that the following assumptions are true about the system hardware and software release that will serve as the base system for incremental change review:

1. The hardware executes instructions correctly.

2. All compilers and assemblers generate code that correctly represents the semantics of each statement.

3. The source code of the program used to generate system tapes and that of the loading program are correct.

4. The source code and the object code of software tools of all programs in the rings 0 and 1 operate according to their interface specifications.

5. The source code and the object code of software tools for reviewing and verifying software changes are correct.

6. All other programs in the system libraries are void of malicious side effects. (a side effect is an action that is not part of the design specification.) The effects of interest here are those that lead to system permissible but unwanted information leakages.

Given the above assumptions, Multics provides several software tools for enumerating software changes and verifying software consistency:

```
check_mst
compare_ascii
compare_mst
compare_object
cross_reference
generate_mst
print_bind_map
print_link_info
```

Complete information on the above commands can be found in the *Multics Commands and Active Functions* manual or the *Administration, Maintenance, and Operations Commands* manual.

The following is a suggested procedure for enumerating and verifying the software changes in a new system release:

1. Do not reload the system release, but rather retrieve it into a temporary library.

2. Compile or assemble (and in some cases macro expand) the security kernal source code. Refer to the microfiche listings for appropriate compiler version and options.

3. Compare the resulting object with the supplied object. This ensures that the supplied object code was generated from the supplied source code. For each bound segment, new bind archives should be created from the new object segments. These should be bound and compared to the supplied bound segments.

4. Generate a system tape. The tape is made by scanning the header file, which lists the names and attributes of all of the objects that go on the tape. Compare the tape to the distributed one.

   Assuming everything was successful, at this point one has verified that the release is, at least, self-consistent, i.e., all individual object segments, bound segments, and system tapes correspond to the source code provided. The next step is to examine the source code changes contained in the new release. The object here is to ensure that no security assertion has been compromised and that the integrity of the release is preserved.

5. Run ASCII comparisons of old source code against shipped source code to produce listings that will be reviewed for security implications.

If no inconsistencies are found in steps 3 and 4 and no changes adversely affecting security are found in step 5, then the release can be safely installed in the normal manner. Changes to the software change verification tools themselves should be reviewed in the same manner, using old versions.

OPERATOR IDENTIFICATION AND AUTHENTICATION

You can require operators to identify themselves (sign_on) before entering commands at the Initializer terminal or the bootload console. The following procedures must be followed:

1. The require_operator_login installation parameter must be set to "yes".

2. At user registration, all users to act as operators must be assigned the "operator" password flag.

3. Operators must use the sign_on command to communicate with the Initializer process from an Initializer terminal or the bootload console. Operators must use the sign_out command to terminate the session. The sign_on command requires operators to specify a valid person_id and password.

If the require_operator_login installation parameter is set to "no", then operators are not required to sign on. If an individual was not registered with the "operator" password flag, then that individual cannot use the sign_on command.

The operator_inactivity_limit installation parameter can be used to specify an inactivity timeout for operators.

SUPERVISING OPERATORS

Multics is designed to prevent operators from compromising system security during normal operation. Once the system has been booted, operators are prevented from directly manipulating secured information. This does NOT mean that incompletely trusted operators can be left unattended to supervise a Multics system. Physical control by trusted people is always required for the following:

- the system's physical storage media (system tapes and disk packs)
- the bootload console at bootload and during crash recovery, until the Answering Service is initialized

Since the system may crash (or be crashed by the operator) at any time, this requires that a fully trusted supervisory person be available at all times when persons who are not fully trusted are present in the machine room.

Furthermore, site procedures must include the following:

- Noncleared operators are not to initiate crash recovery.
- Cleared persons must always supervise crash recovery.
- After every system shutdown and/or crash, system security personnel must verify that a cleared person supervised the recovery. If no cleared person supervised the recovery, then there is a risk of a security penetration.

VOLUME BACKUP LIMITED SERVICE SUBSYSTEM

The volume backup limited service subsystem (LSS) lets you recover from most disk failures while the system is running and available for users. To use the volume backup LSS, the following user identities must be registered:

- Volume_Dumper.Daemon
- Volume_Retriever.Daemon
- Volume_Reloader.Daemon

The above special user identities must be registered with the "multip" and "daemon" attributes. For information on registering the above special user identities, see Section 33. Note that at sites running AIM, these person_ids must be set at system_high and the home directories must be precreated at system_high.

To set up the volume backup system, log in the Repair SysDaemon, or, if running in special session using the Initializer, execute the following command:

```
ec >tools>setup_volume_reloader
```

This exec_com creates all directories, segments and message segments necessary for running the volume backup system. This exec_com also sets suggested access on the directories and segments created. Not all the access that is set is required. If a site wishes, the access created for *.SysMaint.* and *.SysAdmin.* may be removed.

You must also copy the vbk_start_up.ec from the tools directory and use it as the project_start_up.ec for the Daemon project. The command for this procedures is:

```
copy >t>vbk_project_start_up.ec >udd>DAEMON>project_start_up.ec
```

You can then set access on the segment, as appropriate.

The necessary command tables (volume_dumper.ct, volume_retriever.ct, and volume_reloader.ct) are supplied with the operating system software. These command tables restrict the volume backup command set as follows.

The LSS command table for the volume dumper (Volume_Dumper.Daemon) restricts its available command set to:

```
complete_volume_dump
consolidated_volume_dump
delete_volume_log
display_pvolog
display_volume_log
dmpr_unlock_pv
end_volume_dump
incremental_volume_dump
merge_volume_log
preattach_dump_volumes
purge_volume_log
rebuild_pvolog
recover_volume_log
set_volume_log
set_volume_wakeup_interval
verify_dump_volume
volume_cross_check
volume_dump_trace_off
volume_dump_trace_on
wakeup_volume_dump
```

The LSS command table for the volume retriever (Volume_Retriever.Daemon) restricts its available command set to:

```
list_retrieval_requests
retrieve_from_volume
```

The LSS command table for the volume reloader (Volume_Reloader.Daemon) restricts its available command set to:

```
display_volume_log
merge_volume_log
recover_volume_log
reload_volume
verify_dump_volume
```

In addition, all three LSS command tables will allow these commands:

```
exec_com
help
home_dir
logout
system
user
```

All the commands listed above are described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64, with the exception of list_retrieval_requests, exec_com, help, home_dir, logout, system, and user, which are documented in the *Multics Commands and Active Functions* manual, Order No. AG92. (Note that "help" and "logout" here refer to the standard Multics commands by those names, not the initializer commands by those names.)

For more detailed information on volume dumping, see the *Multics System Maintenance Procedures* manual, Order No. AM81.

HIERARCHY BACKUP DAEMON LIMITED SERVICE SUBSYSTEM

The Multics hierarchy backup system protects against the destruction of information maintained by the Multics storage system. The hierarchy backup system preserves recent copies of all segments and directories known to the storage system on magnetic tape, and recovers these copies when needed.

The necessary command table (hierarchy_dumper.ct) and the necessary project start_up.ec is supplied with the operating system software. However, you must copy the sysdaemon_project_start_up.ec from the tools directory and use it as the project start_up.ec for the SysDaemon project. The commands to perform this procedure are:

```
copy >t>sysdaemon_project_start_up.ec >udd>sd>project_start_up.ec
sa >udd>sd>project_start_up.ec r * -replace
```

Within this LSS, there is one LSS command table for the two hierarchy dumpers (Backup.SysDaemon and Dumper.SysDaemon). This command table restricts the hierarchy dumpers' available command set to:

```
backup_cleanup
catchup_dump
complete_dump
end_dump
start_dump
wakeup_dump
```

and:

```
exec_com
help
home_dir
logout
system
user
```

All the commands listed above are described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64, with the exception of list_retrieval_requests, exec_com, help, home_dir, logout, system, and user, which are documented in the *Multics Commands and Active Functions* manual, Order No. AG92. (Note that "help" and "logout" here refer to the standard Multics commands by those names, not the inititalizer commands by those names.)

For more detailed information on the hierarchy backup LSS, see the *Multics System Maintenance Procedures* manual, Order No. AM81.

PART VII
MANAGING I/O DAEMONS

SECTION 27

UNDERSTANDING I/O DAEMONS

The software that handles printing, punching, and card input is called the I/O daemon. The I/O daemon normally runs with highly privileged access on the SysDaemon project. The I/O daemon is organized into a "coordinator" process and a number of "driver" processes; a driver is associated with each local or remote device.

COORDINATOR PROCESS

The coordinator routes messages between one or more daemon drivers and one or more terminals being used as operator consoles, and allows the daemon drivers to run without attached terminals.

DRIVER PROCESSES

The I/O daemon driver processes perform data transfer functions to/from the specified devices. (The I/O daemon processes are said to "drive" the device.) The standard driver modules provided by the system are:

- printer driver module - used for standard Multics printers
- punch driver module - used for standard Multics card punches
- reader driver module - used for standard Multics card readers
- spool driver module - used for major devices that are used to write user print requests onto tape instead of the printer.
- remote driver module - used for all printer/punch/reader stations.

A description of each of the above modules is given in Appendix A.

SECTION 28

SETTING UP THE I/O DAEMON

To create the I/O daemon, you must:

1. Register the IO.SysDaemon "user"
2. Set up the I/O daemon tables
3. Create the I/O daemon queues
4. Log in the Coordinator and driver processes

A description of each procedure is given below.

REGISTERING IO.SYSDAEMON

The I/O daemon (IO.SysDaemon) is automatically registered and installed during initialization of a new Multics site. The IO person_id is registered in the User Registration File (URF), the Person Name Table (PNT), and the Project Definition Table (PDT). The SysDaemon project is registered in the System Administration Table (SAT).

The example below is a sample SAT for a SysDaemon project. The second example below is a sample PDT entry for a SysDaemon project.

```
projectid:        SysDaemon;
projectdir:       >user_dir_dir>SysDaemon;
maxprim:          100;
attributes:       primary_line, guaranteed_login,
                  anonymous, nolist, dialok, multip,
                  bumping, brief, vinitproc, vhomedir,
                  nostartup, daemon, igroup, save_pdir,
                  disconnect_ok;
authorization:    "system_high";
ring:             1,5;
alias:            sd;
group:            System;
groups:           IO, BackGrnd;
grace:            2880;
pdir_quota:       2048;
max_foreground:   0;
max_background:   0;
abs_foreground_cpu_limit: 0;
```

```
Projectid:              SysDaemon;
process_group_id:       Bell.SysAdmin.a;
table:                  PDT;
w_dir:                  >udd>sa>a;
max_size:               256;
current_size:           14;
version:                3;
n_users:                12;
project_dir:            >udd>sa>admin;
rate_structure:         default (0);


personid:               10;
 state:                 1;
 now_in:                3;
 n_foreground:          3;
 attributes:            nobump, guaranteed_login, nopreempt,
                        dialok, multip, vinitproc, vhomedir,
                        nostartup, daemon, igroup, save_pdir,
                        disconnect_ok, save_on_disconnect;
 initproc:              >system_library_tools>iod_overseer_;
 homedir:               >user_dir_dir>SysDaemon>10;
 grace:                 2880;
 ring:                  4, 5, 4;
 group:                 10;
 max_foreground:        0;
 max_background:        0;
 abs_foreground_cpu_limit: 0;
 limit:                 open;
 shift_limit:           open, open, open, open, open, open, open, open;
 user_warn_dollars:     10.00;
 user_warn_percent:     10;
 user_warn_days:        10;
 warn_dollars:          10.00;
 warn_percent:          10;
 warn_days:             10;
 dollar_charge:         $2167.63;
```

Process Overseer

The SysDaemon project is assigned a special process overseer called the "iod_overseer_". The iod_overseer_ process overseer is required for the IO.SysDaemon users.

Authorization

System access authorization for a daemon is specified during the registration process. All daemon processes should be given an authorization range of system_high to system_low. This requires that the System Administration Table (SAT) authorizations for the SysDaemon and Daemon projects be set to system_high. Similarly, the Person Name Table (PNT) authorizations for each person (daemon) registered on these two projects must be set in the range system_low to system_high. All daemons have a default login authorization of system_low. The default login authorization of the daemons may be changed by specifying the change_default_auth (-cda) control argument to the operator login command.

Attributes

For proper operation, the I/O daemon requires the assignment in the PDT of the "daemon" and "dialok" attributes.

I/O DAEMON TABLES

To manage the use and operation of the I/O daemon, an administrative data base exists that can be adapted to the specific needs of a particular Multics site. This data base contains several different tables of information and hence is referred to as the "I/O daemon tables". The data is generated from a source language description.

The purpose of the I/O daemon tables source language is to define the devices and the request types to be used by the I/O daemon. A source file consists of a sequence of statements and substatements that define and describe each device and request type. In addition, certain global information items are defined that do not pertain to any particular device or request type.

The creation of the I/O daemon tables is described in the *Multics System Maintenance Procedures* manual, Order No. AM81. A description of the iod_tables source language description is provided in Appendix A.

CREATION AND MAINTENANCE OF I/O DAEMON QUEUES

The I/O daemon queues are created automatically by use of the create_daemon_queues command (described in the Administration, Maintenance and Operations Commands manual, Order No. GB64). The queues are created in the same directory as the I/O daemon tables, i.e., >daemon_dir_dir>io_daemon_dir. The command determines what queues to create based upon information contained in the iod_tables segment. For each request type, one to four queues are created depending on the value of Max_queues or the per request type max_queues, whichever is in effect. The name of each queue is of the form XXX_N.ms where XXX is the request type name and N is the priority number. The ms suffix indicates that each queue is a ring 1 message segment.

LOGIN AND INITIALIZATION OF THE DAEMON COORDINATOR

The step-by-step procedure for logging in and starting up the daemon coordinator is available in the *Operator's Guide to Multics*, Order No. GB61.

LOGIN AND INITIALIZATION OF DEVICE DRIVERS

Step-by-step procedures for logging in and starting up printers, card punches, card readers, and remote devices are available in the *Operator's Guide to Multics*, Order No. GB61.

SECTION 29

MANAGING I/O DAEMON OUTPUT

The I/O daemons are designed to run with little operator intervention; however, prior to their operation you must establish the rates for their use, and set the access privileges for the daemon queues.

SETTING RATES FOR PRINTING AND PUNCHING

The daemon rates are kept in the installation_parms segment; you can inspect the rates and modify them by specifying the queue_prices parameter with the ed_installation_parms command (see the *Administration, Maintenance and Operations Commands* manual, Order No. GB64). There are a maximum of four daemon queues for each request type (see "Resource Price List" below for more information). Default I/O daemon rates per 1000 lines are stored by queue. For example:

Queue	I/O Daemon
1	1.00
2	0.75
3	0.50
4	0.50

The default value which applies to all daemon queues is $1.80 per 1000 lines.

RESOURCE PRICE LIST

The resource price list is a set of named resource prices kept in the installation_parms segment. It is a generalization of the prices described above. Currently it supplements daemon rates, allowing separate prices to be charged for special forms used in I/O daemon requests. The daemon rates, described above, continue to be used by default.

The price names used with the line_charge keyword in the iod_tables segment must be defined in the resource price list before the iod_tables segment can be compiled.

LISTING AND CHANGING EXTENDED ACCESS ON I/O DAEMONS

Because the I/O daemon queues are message segments, access to the queues is determined by extended access modes. Extended access is an additional field of access used with message segments or queues to further restrict operations on a message segment.

Give the IO.SysDaemon identity full extended access, i.e., add, delete, read, own, and status (adros) to all queues. For standard system queues (i.e., queues for which the driver_userid of the corresponding request type is IO.SysDaemon) aros permission is given to all users. Otherwise, the assumption is made that the queues are dedicated to the particular project named in the driver_userid. In this case, aros permission is given just to users of that project.

You can use the list_acl command to list the extended access on the queues, and the set_acl command to change the extended access on the queues as shown in the examples below.

Changes to the I/O daemon tables must sometimes be coordinated with changes to the I/O daemon queues. In particular, when a new request type is added, new queues must be created for this request type. This can be done as soon as the iod_tables segment has been recompiled. Use of the create_daemon_queues command does not affect any existing queues, but does create new queues for any newly defined request types. If a request type is removed from the I/O daemon tables, the queues are not automatically deleted. The delete command can be used to delete obsolete queues.

SECTION 30

MANAGING I/O DAEMON INPUT

MANAGING A CARD INPUT STATION

The configuration and management of a card input station requires the following procedures:

1. Register card input users
2. Register card input passwords for users
3. Register stations and their passwords in the PNT
4. Create an access control segment for a station
5. Grant access to a the ACS for a station

Each of these topics is discussed below. A complete description of card deck formats is described in the *Multics Programmer's Reference Manual*, Order No. AG91.

Registering Card Input Users

For a user to submit a card deck for input to Multics, the following conditions must be met:

1. A user must be registered and assigned a card-input password or given permission to use the null password feature.

2. A special access control segment must exist in the user's mailbox directory. Proper access must be set for the station in order for it to read card decks.

3. The user must have permission to use the card input station. This is granted by you on the ACL of the station access control segment.

For RJE jobs, the tag portion of the process group ID of the absentee process (which is used in access control calculations) is "p". Yourself or a user may deny access to RJE jobs with the ACL term:

```
null *.*.p
```

or similar ACL terms, assuming that there does not exist a more specific ACL term that gives access.

Registering Passwords for Card Input Users

You must assign each user a card input password in the Person Name Table at user registration time for the user to use any form of card input on Multics. The card input password defined should be different from the user's interactive password. The Person_id and password of the user must be provided on control cards at the time the deck is submitted.

If the user was not assigned a card input password when initially registered, add a card input password to the users registration profile by:

1. Issuing the command line: ec master new_user Person_id or ec master change Person_id

2. Adding the card input password for the user.

Section 32 contains a complete description of registering users and changing user registration attributes.

Registering Stations and Their Passwords in the Person Name Table

Register a station and password as you would register any user by:

1. Issuing the register command to register each card input station (central or remote) in the PNT.

2. Associating a card input password with each station.

The station card input password must be specified by the operator as part of the sign-on sequence.

Creating an Access Control Segment for a Station

Each card input station must have an access control segment residing in the directory >system_control_1>rcp. All users allowed to submit card input from a station must be on the ACL of the stations access control segment (e.g., Station_A.acs). The card reading process must have s access to this directory. If access is not specified, or if this segment does not exist, the card input is aborted.

In the following example, an access control segment is created for Station_A, all users associated with Project_A are added to the acs for the station, and the card reader process (Card_Input.Daemon) is granted access to the directory containing the acs for the station.

1. Create the acs for Station_A:

```
cr >sc1>rcp>Station_A.acs
```

2. Assign read and execute access to the station for all users of Project_A:

```
sa >scl>rcp>Sation_A.acs re *.Project_A
```

3. Grant Card_Input.Daemon s access to the directory contained the acs for Station_A:

```
sa >scl>rcp s Card_Input.Daemon
```

Note that in the example above, the star convention is used in the normal fashion, for example:

```
r    user.*.*
re   *.Project_A.*
n    *.*.*
```

This check allows a site to specify that a certain station is reserved for the use of a certain group of users, perhaps those who pay for the equipment. It can also be used to ensure that certain stations are not used to submit RJE card input for privileged users, such as *.SysAdmin, who should never normally use the facility.

Giving Stations Access to Submit Card Input

It is the users' responsibility to explicitly permit a station to submit card input for them by creating the segment "card_input.acs" in their mailbox directory. The ACL for this segment must give r access to each station that the user permits to submit bulk data input and e access for each station that the user permits to submit RJE jobs. For example:

```
re Station_A.*.*
```

The card reading process must have s access to the project_id and person_id directories. If this segment does not exist or if the access is not as specified, the card input is aborted.

UNDERSTANDING PROXY/REMOTE JOB ENTRY

Remote Job Entry refers to the submission of Multics jobs in the form of punched cards to be entered from the card reader. The card images are copied into an absentee input segment in the normal system pool storage used for bulk data input card image segments. When the user's deck has been successfully read, an absentee request is submitted on behalf of the user who provided the deck. A special header is added to the absentee input segment so that a dprint request of the absentee output segment is automatically generated using the request type associated with the remote terminal or the request type of the local printer, depending on the input device.

The example below is the required format for a card deck for remote job entry.

```
++RJE  DECK_NAME  PERSON_ID  PROJECT_ID
++PASSWORD  PASSWORD
++AIM  ACCESS CLASS OF ABSENTEE PROCESS
++RJECONTROL  CONTROL ARGS TO THE EAR COMMAND
++RJEARGS  ARGUMENTS FOR THE ABSENTEE PROCESS
++EPILOGUE  COMMAND
++FORMAT PUNCH_FORMAT MODES
++INPUT
   .
   .
(user absentee file)
```

The only cards required are the first which is an identifier card, the second which is a password card and the last which is the end of control input. For an explanation of all the control cards refer to *Multics Programmer's Reference Manual*, Order No. AG91.

The user should submit a complete card deck to operations.

SETTING ACCESS TO A CARD INPUT STATION FOR RJE SUBMISSION

Any process (e.g., IO.SysDaemon and Card_input.Daemon) which is to read and process remote job entry (RJE) card input must have e access to the segment:

```
>system_control_1>proxy>absentee_proxy.acs
```

to be able to submit proxy absentee requests.

READING CARDS

Remote terminals with card readers and the central site reader can be used to enter user card decks for both bulk data input and RJE. The operator must issue the read_cards command to the card reading process to start reading in user card decks. The card reading process may be either the central site Card_input.Daemon (see "Reading Cards at the Central Site" below) or an I/O daemon driver.

For a complete description of the card formats required for bulk data input or remote job entry, see the *Multics Programmer's Reference Manual*, Order No. AG91.

UNDERSTANDING THE CARD POOL

The card pool is a directory (usually >ddd>cards) where temporary directories are saved that contain segments of the card images that are read in from the card reader. The temporary directories have the same names as the personids that created them. It is the responsibility of the users to copy their directories into their home directories.

UNDERSTANDING THE STRUCTURE OF THE CARD POOL DIRECTORY HIERARCHY

The storage pool for card deck image segments consists of a subtree of the directory hierarchy, which is headed by >daemon_dir_dir>cards. One access class directory for each access class is contained in the cards directory. Storage is always allocated within the access class directory that corresponds to the caller's authorization. Person directories are contained in the appropriate access class directory. A person directory is created for each person who needs temporary storage. A person directory contains all segments and multisegment files for a given person at a given access class. For example, if a user with Person_id TSmith is at system_low, the following directory is allocated for his card deck image segments:

```
>daemon_dir_dir>cards>system_low>TSmith
```

MANAGING THE CARD POOL

Card pool management requires that you perform the following actions to ensure the proper operation of the card pool:

- Set the card pool quota
- Set the proper access
- Perform periodic cleanup of the card pool

A description of each of these procedure is given below.

Managing the Card Pool's Quota

The quota you assign to a card pool directory is dependent upon the amount of card input used at the site, the number of users expected to use the card input facility, and the the approximate size of the jobs submitted by the card input users. Therefore, a certain amount of trial and error is involved in quota assignment until you can determine the exact needs of your card input community.

Setting Access to a User Directory in the Card Pool

In the card pool directory, user directories are created automatically as required. As user card image segments are entered into the card pool, temporary directories are created for the users to temporarily contains the card image segments until the user can copy the card image segment into his or her own directory.

Doing Periodic Cleanup of the Card Pool

The clean_card_pool command lets you delete inactive card image segments (in >ddd>cards) created by the system card reading process that are older than a specified number of days.

After the pool is cleaned, all empty person directories and access class directories are deleted, all links and directories contained in a person directory are deleted regardless of age, and all links and segments in an access class directory are deleted regardless of age. For a complete description of the clean_card_pool command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

SECTION 31

RUNNING I/O DAEMONS

UNDERSTANDING ADMINISTRATIVE EXECUTION COMMAND FILES

An I/O daemon admin exec_com must be written by a you to provide site-defined driver x command functions. The x command allows drivers to execute an admin exec_com on a site-defined basis. The use of admin exec_coms is optional, but when missing, the driver x command will not work. For detailed information on the x command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

All I/O daemon admin exec_com segments must be located in the >ddd>idd directory and follow standard exec_com rules. There are two types of admin.ec files:

- general - iod_admin.ec
- device specific - <device>_admin.ec

These differ only in segment name to allow the site to separate x command functions by device name (station_id for remote stations). The general exec_com (iod_admin.ec) is used by any driver that cannot find a device-specific exec_com.

A <device>_admin.ec segment is a device-specific exec_com for the given major device; for example, prta_admin.ec is specific to device prta. Added names can be used to group several devices under a single device-specific exec_com.

The Multics command iod_command may be used within an admin exec_com to execute arbitrary I/O daemon commands. For example:

```
iod_command defer_time 30
```

may be used in an admin exec_com to change the auto defer time limit for the current driver to 30 minutes. The iod_command command is described in detail in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

When writing an I/O daemon admin exec_com, the process that executes it will, most likely, have full SysDaemon access and privileges to the system. Therefore, care must be given in choosing what functions should be placed at the hands of a remote station operator or an inexperienced device operator.

Sample I/O Daemon Administrative Command File

In the following sample iod_admin.ec, responses/actions are configured for the x command functions cdr, car, pq. A simple help facility is built into the iod_admin.ec and a command "catchall" is specified in case the operator enters a command not defined in the ec.

```
& ------------------------------------------------------
&
& iod_admin.ec    (to be found in >ddd>idd)
&
& This is the exec_com for the IO Daemon driver "x" command.
& The first argument to the "x" command is &1 in this exec_com.
& The standard action is to transfer control to a label
& which will implement the function of &1.
&
& Any arguments associated with an "x" command function begin
& with &2 in this exec_com.

&command_line off
&goto &1.command

&label help.command
&
& For "x help" print a list of x command functions.
&
&print cdr -user Pers.Proj <seg_ident>
&print car -user Pers.Proj <seg_ident>
&print pq {ldr_args}
&quit

&label cdr.command
&
& For "x cdr -user Pers.Proj <seg_ident>"
$ to cancel a dprint request for this driver
&
&if [not [exists argument &2]]
&then &goto missing_arg.error
cdr -rqt [iod_val request_type] &f2
&quit

&label car.command
&
& For "x car -user Pers.Proj <seg_ident>"
& to cancel an RJE job sent by this station
&
&if [not [exists argument &2]]
&then go to missing_arg.error
car -sender [iod_val station] &f2
&quit
```

```
&label pq.command
&
& For "x pq {ldr_args}"
& to list all requests that can be processed by this driver
&
&if [exists argument &2]
&then ldr -a &f2
&else ldr -a -admin -rqt ([iod_val rqt_string]) -tt
&quit

&label &1.command
&
& This is a catchall for any undefined command functions.
&
&print Undefined driver x command function.
&
ioa_ "received command: ^(^a ^)" &f1
&
&quit

&label missing_arg.error
&
&print Expected argument missing. Try again or type "x help".
&
&quit
```

Sample Driver I/O Daemon Administrative Command File

The following is a sample driver command file for a NEC5525 printer:

```
& nec_admin.ec  Special commands for the NEC I/O driver
&
&
&command_line off
&goto command.&1
&
&
& "ldr_text" command: List all text queues
&    Usage: x ldr_text
&
&label command.ldr_text
&
format_line "Quality print queues at ^a:^a on ^a ^a ^a:" [picture 99
  [hour]] [picture 99 [minute]] [day] [month_name] [long_year]
&
list_daemon_requests -request_type (pica_10 courier_10 focus_10
  elite_12 prestige_12 math_12 greek_12) -user *.* -pathname -all
&
&quit
&
&
& "log_init" command: Initializes logging of requests by Message Coordinator
&    Usage: x log_init
```

```
&
&label command.log_init
&
&print This command is no longer used.
&quit
&if [not [io attached log_i/o]] &then &goto log_init.setup_switches
&if [equal [io io_module log_i/o] broadcast_] &then &goto log_init.in_use
&if [equal [io attach_desc log_i/o] "syn_ user_i/o"] &then &goto
log_init.detach_and_setup &
&print x log_init: log_i/o attached through unknown I/O module.
&quit
&
&label log_init.detach_and_setup
io detach log_i/o
&label log_init.setup_switches
io attach log_mr mr_ spbn
io open log_mr stream_output
iocall attach log_i/o broadcast_ log_mr
iocall attach log_i/o broadcast_ user_output
&quit
&
&label log_init.in_use
&print x log_init: This request should only be used once per process.
&quit
&
&
& "reply" command: Allows the operator to send arbitrary commands
& to the driver's master terminal from the slave terminal
&
&label command.reply
&
sm -pn >udd>SysDaemon>SIPB_print>spbd &rf2
&quit
&
&
& "defaults" command: Sets up all the default parameters; this
& is usually done automatically, but sometimes needs to be redone
&
&label command.defaults
&
iod_command auto_start_delay 3600
iod_command pause_time 0
iod_command slave_term nolog
iod_command slave_term modes ^lower_case
&
iod_command defer_time pica_10 30
iod_command defer_time elite_12 30
iod_command defer_time focus_10 30
iod_command defer_time courier_10 30
iod_command defer_time prestige_12 30
iod_command defer_time greek_12 30
iod_command defer_time math_12 30
```

```
&
iod_command prt_control pica_10 autoprint
iod_command prt_control elite_12 autoprint
iod_command prt_control focus_10 autoprint
iod_command prt_control courier_10 autoprint
iod_command prt_control prestige_12 autoprint
iod_command prt_control greek_12 autoprint
iod_command prt_control math_12 autoprint
&quit
&
&
& "ready" command: Does what the standard daemon "ready" command does,
& but makes sure that only one device is ready at a time
&
&label command.ready
&
iod_command halt -all
iod_command ready &r2
& check device name against known devices.
&if [not [or [equal "&q2" (pica_10 courier_10 focus_10 elite_12 prestige_12
    greek_12 math_12)]]]
&then &print Warning: unrecognized device "&q2".
&quit
&
&
& "hangup" command:  Prepares the daemon for the plug to be pulled
& on the terminal
&
&label command.hangup
&
iod_command logout
&quit
&
&
&
&label command.reinit_print_data
initiate >udd>SysDaemon>SIPB_print>print_data_ -force
&quit
&
&
&label command.display_print_data
&label command.dspd
display print_data
&quit
&label command.&1
&
exec_com &ec_dir>iod_admin &rf1
&quit
```

PART VIII
MANAGING PROJECTS AND USERS

SECTION 32

PROJECT REGISTRATION

Registering a new project on Multics, or changing the environment characteristics of an existing project, is accomplished with the interactive project registration commands. However, the greater part of the registration process includes planning the environment for a project and the users associated with the project. To do this, you should be thoroughly familiar with the environment characteristics that can be assigned to a project.

ORGANIZING PROJECTS ON THE SYSTEM

Projects are structured on the system in a hierarchical fashion from the >udd directory which is directory subordinate to the root ">".

PROJECT REGISTRATION FILES

When you register a project, the keywords that define the project's environment characteristics are located in the following system files:

- System Administration Table (SAT)
- Project Master File (PMF)

In addition to the above files, a "projfile" and a "reqfile" exist that contains entries for each registered project.

Before beginning a discussion of project registration, you should be familiar with the descriptions of the Project Master File and System Administration Table given below.

UNDERSTANDING THE PROJECT MASTER FILE

The Project Master File contains keywords that define the environment for the users associated with the project. The PMF is created automatically during the initial registration process and initially assigned default keyword values (see Table 1-1). The default keyword values can be subsequently edited to add or change project keyword assignments, add new users to the project, or change the keyword assignments of existing users.

For system use, the PMF must be converted to a binary table called the Project Definition Table (PDT) and installed in >sc1>pdt.

Once a project is registered, you can view the contents of the project's PMF by using the print_pdt command. For a complete description of this command, see "Viewing the Contents of the Project Master File".

Project Master File Format

The keywords in a Project Master File apply to either the project or the users associated with the project. To distinguish between a project keyword and a keyword for a specific user, they are referred to as either:

- global keyword - Defines project environment characteristics and begins with an *uppercase* letter.
- local keyword - Local keywords define user characteristics and override the global keyword counterpart specified for the entire project. Local keywords begin with a *lowercase* letter.

Global keywords specified in the PMF apply to all users associated with the project. If you specify a local keyword for a user that is different from the global keyword specified for the project, the local keyword overrides the global keyword specified for the project and the user is assigned the environment characteristic specified by the local keyword. This is how you can tailor the environments of specific users. However, as shown in Table 31-3, there are many PMF values that have corresponding SAT entries for security purposes; any keyword value specified in the SAT overrides both local and global PMF keyword values.

Keywords

Projectid: character_string;

where character_string is the name of the project (Project_id). This name must begin with a capital letter and must be the same as the project name in the SAT. The Project_id must be the first keyword entry in the PMF.

personid: character_string;

where character_string is a Multics Person_id or an asterisk (*). An asterisk indicates an anonymous use (see "User Registration for more information on anonymous users). This keyword begins a new user entry.

password: character_string;

where character_string is the password used by anonymous users when they log in. This keyword can only be specified for an anonymous user entry and should not be confused with the password assigned to registered users.

homedir: pathname;
Homedir: pathname;

where pathname is the absolute pathname of the user's home directory at login. If the user is to have no permanent storage in the storage system hierarchy, this pathname may have the form " [pd] >name", and the home directory is then created for the user at login as a subdirectory of his process directory.

initproc: pathname;
Initproc: pathname;

where pathname is the absolute or relative pathname of the user's process overseer procedure. If the pathname is followed by a comma and the string "direct", the overseer is not called by the init_admin_ subroutine, but is invoked directly from ring 0 at process creation time; this usage requires very special programming in the overseer procedure, but is useful for finely tuned environments. See the *Multics Programmer's Reference Manual*, Order No. AG91 for a detailed description of process overseer procedures.

environment characteristics: character_strings;
Attributes: character_strings;

where character_strings are attribute names separated by commas. The named environment characteristics are turned on for the user's). If the attribute name is preceded by a circumflex (^), the attribute is turned off for the user's) even if a default or global characteristic specifies it on. (See "Keyword Default Values" below.) Default environment characteristics are completely replaced each time an Attributes statement is encountered. However, an environment characteristics statement does not replace the default environment characteristics by those specified in the statement; it assigns the user the logical OR of the default environment characteristics with those specified in the statement. Thus, it is necessary to explicitly turn off any of the current default environment characteristics that a particular user is not to have. Valid environment characteristics are:

null, none
: may be used as the only value in an Attributes statement, to turn off all default environment characteristics; illegal in an environment characteristics statement

nobump
: user not subject to preemption by anyone

guaranteed_login
guar
: user may use the -force control argument to the login

nopreempt
: user not subject to preemption by others in group

nolist
: user should not be listed in whotab; users with this attribute are not listed by the who command

dialok
dial
: user may accept dial requests

multip
multi_login
: user may log in more than one process

preempting
bumping
: user may preempt others in same load control group

brief
: user is permanently in brief mode (as if -brief had been given in login command), and does not receive the messages associated with a successful login

vinitproc
v_process_overseer
: user may specify a process overseer or outer module on the login command line; user may also replace his process overseer, outer module, or other procedures by placing a copy in his home directory, because the home directory is in the search path during process initialization for a user with this attribute.

vhomedir
v_home_dir
: may specify home directory at login

nostartup
no_start_up
: user may escape from using his start_up.ec

no_secondary
no_sec
: user may not have secondary load-control status

no_primary
no_prime
: user may not have primary load-control status

op_login
daemon
: user may be logged in by operator, via message coordinator

no_warning
nowarn
: user is permanently in no_warning mode (as if -no_warning had been given in login command) and never receives system warning messages

igroup
: user can be in an individual load control group that is different from the project's default load-control group

save_pdir
: save process directory after fatal process error; used for debugging purposes at development sites

disconnect_ok
: user may have saved disconnected processes, i.e., user may use the -save argument to the login command. This attribute must also be on in the project's SAT entry, to give users on the project permission to use -save.

save_on_disconnect
save

user's process is saved on disconnection, by default; relieves user from typing -save_on_disconnect at each login; can be overridden by typing -no_save_on_disconnect at login time. If this attribute is on (either individually or because of an Attributes statement) then the disconnect_ok attribute is forced on for the user (but the disconnect_ok attribute must also be on in the project's SAT entry to give users on the project permission to have saved disconnected processes). If the system administrator turns on this attribute in the project's SAT entry, then disconnected processes of all users on the project are saved by default, regardless of the setting of this attribute in the PDT.

grace: decimal_integer;
Grace: decimal_integer;

where decimal_integer is the number of minutes after login for which the user is protected from preemption by other users.

group: character_string;
Group: character_string;

where character_string is the name of a load control group. The group or Group statement is used to define the project's default load-control group.

outer_module: character_string;
Outer_module: character_string;

where character_string is the name of the terminal device outer module of a user with an interactive process. (The statement has no effect for absentee processes.)

abs_foreground_cpu_limit: decimal_integer;
Abs_foreground_cpu_limit: decimal_integer;

where decimal_integer is the upper CPU time limit in seconds for the user's foreground absentee jobs. A value of zero means no limit.

max_background: decimal_integer;
Max_background: decimal_integer;

where decimal_integer is the maximum number of concurrent background (absentee) processes that the user may have. A value of zero means no limit.

max_foreground: decimal_integer;
Max_foreground: decimal_integer;

where decimal_integer is the maximum number of concurrent foreground processes that the user may have. A value of zero means no limit.

pdir_quota: decimal_integer;
Pdir_quota: decimal_integer;

where decimal_integer is the quota to be placed on the user's process directory. If the value is zero, or if no quota is specified, the system default process directory quota is placed on the user's process directory.

limit: floating-point_number or "open";
Limit: floating-point_number or "open";

where floating-point_number is the absolute dollar limit that the user may spend in any monthly billing period. If the user exceeds this limit, he is automatically logged out and is unable to log in until the end of the month, or until the limit is changed and a new PDT installed. If no limit value is imposed, the character_string "open" is used.

shift_limit: floating-point_numbers or "open";
Shift_limit: floating-point_numbers or "open";

where floating-point_numbers are the user's interactive resource limits, separated by commas, on shifts 1, 2, 3, 4, 5, 6, 7, and 0. Shifts not listed receive an "open" limit. Absentee and I/O charges, although counted against the total expenditure limit, are not counted against the shift limits. (Shift 0 is listed last, because most installations do not define a shift 0.)

cutoff: limit {,date {,increment}};
Cutoff: limit {,date {,increment}};

where:

limit is a floating-point number specifying a dollar value, or the word "open" or no limit.

date is a date in a form acceptable to the convert_date_to_binary_ subroutine, or "now", "open", or "never".

increment
increment is one of the following keywords:

```
never    - do not reset
daily    - reset every day
monthly  - reset to beginning of next month
yearly   - reset to one year later
cyear    - reset to beginning of next year
fyear    - reset to July 1 of next year
```

The cutoff statement is used to set a cutoff limit on the user; this is different from the monthly spending limit specified by the limit statement (above). The cutoff statement has three different interpretations, depending on whether one, two, or all three arguments are given.

If just the limit argument is given, then an absolute spending limit is imposed. When the user's spending reaches this limit, he is logged out and not allowed to log in again until a new PDT, containing a higher cutoff limit for that user, is installed. The cutoff limit is not reset by monthly billing, and is imposed in addition to the monthly limit. The word "open" causes no cutoff limit to be imposed; this is the default (in the absence of a Cutoff statement).

If just the limit and date arguments are given, then in addition to the absolute spending limit described above, a cutoff date is imposed. When that date and time arrives, the user is logged out and not allowed to log in again until a new PDT, containing a later cutoff date for that user, is installed. The word "open" causes no cutoff date to be imposed; this is the default (in the absence of a Cutoff statement).

If all three arguments are given, then the meanings of the first two arguments are changed, and a periodic spending limit is imposed. The increment argument specifies the time period after which the user's spending against this limit should be reset to zero. The limit argument specifies how much the user is allowed to spend during the time increment. The date argument specifies the date and time at which the first time period is to end. It is used only at the time the PDT is installed. Each time the cutoff date arrives, a new cutoff date is set, as specified by the increment argument, and the user's spending against the cutoff limit is reset to zero. When this third type of cutoff limit is imposed, the user is logged out as soon as his spending exceeds the cutoff limit, and he is not allowed to log in until the cutoff date arrives.

NOTE: It is possible to predate the date argument. When the user next logs in, the periodic spending limit is reset, and a new cutoff date is established based on login time and the specified increment.

warn_days: decimal_integer;
Warn_days: decimal_integer;

where decimal_integer is the day threshold for cutoff warning messages. When the project's account has fewer than this many days remaining until its cutoff date, the user receives a warning message to this effect at login time.

warn_percent: decimal_integer;
Warn_percent: decimal_integer;

where decimal_integer is a number in the range 0 through 100, specifying the percent threshold for cutoff warning messages. When the project's account has less than this percent of its funds remaining, the user receives a warning message to this effect at login time.

warn_dollars: floating-point_number;
Warn_dollars: floating-point_number;

where floating-point_number is the dollar threshold for cutoff warning messages. When the project's account has less than this amount remaining, the user receives a warning message to this effect at login time. This keyword applies to the project's account, not the individual user's spending limit.

user_warn_days: decimal_integer;
User_warn_days: decimal_integer;

where decimal_integer is the day threshold for cutoff warning messages. When the user's account has fewer than this many days remaining until absolute cutoff (cutoff increment is "never"), the user receives a warning message to this effect at login time.

user_warn_percent: decimal_integer;
User_warn_percent: decimal_integer;

where decimal_integer is a number in the range 0 through 100, specifying the percent threshold for cutoff warning messages. When the user's account has less than this percent of funds remaining, the user receives a warning message to this effect at login time. This warning may be triggered by any of the specified fields, limit, shift_limit, or cutoff.

user_warn_dollars: floating_point_number;
User_warn_dollars: floating_point_number;

where floating_point_number is the dollar threshold for user spending warning messages. When the user's account has less than this amount remaining, the user receives a warning message to this effect at login time. This warning may be triggered by any of the specified fields, limit, shift_limit, or cutoff.

subsystem: pathname;
Subsystem: pathname;

where pathname is the pathname of a Multics subsystem.

ring: decimal_integer1, decimal_integer2 {,decimal_integer3};
Ring: decimal_integer1, decimal_integer2 {,decimal_integer3};

where decimal_integer1 and decimal_integer2 define the minimum and maximum values for the ring in which the user can create a process. decimal_integer3 specifies the default initial ring at login. All values must be in the range 1 through 7. decimal_integer1 must be less than or equal to decimal_integer2. decimal_integer3 must lie in the range defined by decimal_integer1 and decimal_integer2. If the value of decimal_integer3 is not specified, decimal_integer1 is used.

authorization: aim_range;
Authorization: aim_range;

This statement specifies the range of access authorizations at which the user may log in. An access authorization range is of the form:

```
low_auth:high_auth
```

If commas are used (to specify categories), or spaces (to improve readability) the entire range must be enclosed in quotation marks.

If neither an Authorization or any authorization statements are present in the PMF, the default authorization is the minimum login authorization allowed to the project by the System Administrator.

lot_size: decimal_integer;
Lot_size: decimal_integer;

where decimal_integer is the size of the user's linkage offset table (LOT). The LOT is a per-ring array of pointers that point to the linkage information for each procedure segment known in the given ring.

kst_size: decimal_integer;
Kst_size: decimal_integer;

where decimal_integer is the number of segment numbers that are allocated for the user's process in the known segment table (kst).

cls_size: decimal_integer;
Cls_size: decimal_integer;

where decimal_integer is the size of the user's initial combined linkage region.

Sample Project Master File

```
Projectid:          Foo;
Attributes:         vinitproc, vinitproc, vhomedir, nostartup,
                    save_on_disconnect, disconnect_ok;
Ring:               4, 5, 4;
Authorization:      "system_low:Secret";

personid:           User1;
 attributes:        dialok;
 initproc:          >udd>Foo>special_process_overseer_;
 ring:              4, 5, 4;

personid:           User2;
 ring:              5, 5, 5;
 authorization:     system_low;

personid:           User3;

personid:           User4;
 authorization:     system_low;

end;
```

MODIFYING THE PROJECT MASTER FILE

The method used to edit the Project Master File depends on whether the project is delegated or undelegated. If the project is delegated, the Project Administrator is responsible for editing the PMF as described below in "Adding Users and Changing the Default Keywords of a Delegated Project". If the project is undelegated, you are responsible for editing the PMF as described below in "Adding Users and Changing the Default keywords of an Undelegated Project".

UNDERSTANDING THE PROJECT DEFINITION TABLE

The Project Definition Table (PDT) is the binary, machine-usable version of Project Master File. You must convert the PMF into the system Usable PDT and then install the PDT as described below.

Converting and Installing the Project Master File

Once a project is registered, or the keywords of an existing PMF have been modified, the PMF must be converted to its binary version for system use. The binary version of the Project Master File is called the Project Definition Table (PDT). If the master.ec is used to register projects, conversion and installation is performed automatically. For delegated projects, the master.ec cannot be used and the PMF must be converted with the cv_pmf command and installed with the install command, as described below.

The syntax of the cv_pmf command is:

```
cv_pmf project.pmf
```

Following conversion, the PMF must be installed via the install command. The syntax of the install command is:

```
install project.pdt
```

You have the option to issue these commands manually each time a new project is registered or the keywords of an existing project have been changed, or use the "master.ec" described above which automatically converts the PMF and installs the PMF as required. It is recommended that the master.ec be used when registering a new project or changing the environment characteristics of an existing undelegated project.

Viewing the Contents of the Project Master File

The contents of the Project Master File can be printed in readable form with the print_pdt command. This allows you to view all of the data in the PMF and determine if any changes are required. If changes are required, you can then make the changes (see "Editing the Project Master File").

For a complete description of the print_pdt command see the *Administration, Maintenance, and Operations Commands manual*, Order No. GB64.

PROJECT REGISTRATION COMMANDS

As a system Administrator, the commands necessary to register new projects or change the characteristics of existing projects are:

- {ec master} new_proj – the interactive command that lets you register a new project on the system
- ec master pmf – lets you edit the Project Master File for an "undelegated" project (i.e., a project not assigned a project administrator).
- {ec master} edit_proj – an interactive command that lets you edit the keyword assignments in the SAT for an existing project.

In addition, there are several other project related commands that allow you to delete projects, change project delegation, display various project registration files, etc. For details on these commands, refer to the Administration, Maintenance and Operations Commands manual. Order No. GB64.

The registration commands, new_proj and edit_proj, should be used in conjunction with the "master.ec" described below. When these commands or any other command is used in conjunction with the master.ec, they are called "entry points" because they are now part of the master.ec.

The command line "ec master pmf" can be used only to edit "undelegated" projects (see "Editing an Undelegated Project's Project Master File").

The master.ec is located in >udd>SysAdmin>lib. It provides extensive argument checking and automatically converts a PMF to its binary version, the PDT, and installs a new or edited Project Master File in the appropriate place in the system. When the master.ec is used with the new_proj command, it also automatically creates a PMF for the project and assigns default environment characteristics (keywords) to the project. In addition, the master.ec prevents simultaneous editing of a project file by another person.

When using the master.ec with the project registration commands, change to the directory: >udd>SysAdmin>admin and precede the command_entry_point with "ec master" as follows:

```
ec master command_entry_point
```

If you invoke the project registration command "new_proj" instead of "ec master new_proj" only the new_proj command is executed. If "ec master new_proj" is typed, the PMF is automatically converted and installed.

REGISTERING A PROJECT WITH THE NEW_PROJ COMMAND

Prior to registering a new project, all users to be associated with the project should be registered on the system and have valid Person_ids assigned to them (see User Registration). Also, you should know which environment characteristics (keywords) you want to assign to the project. The keywords displayed by the new_proj command are defined below; PMF keywords are described in detail in "PMF Format".

The new_proj command prompts you for the keywords required to initially register a project and, when used with the master.ec, automatically creates the PMF for the new project with default keyword values. Entries are also made to the projfile and reqfile. The new_proj command *does not* prompt for PMF entries. Note that any values you specify in the SAT entry for the project, take precedence over values specified for the project in the PMF.

The format of the new_proj command is:

```
{ec master} new_proj project_name
```

The new_proj command displays the prompts necessary to initially register a new project as shown in the example below. On the "first pass", the new_proj command does not prompt for all of the keywords that can be associated with a project. Instead, default values are assigned to the remaining keywords for which no prompts were issued. However, if the System Administrator replies "yes" to the prompt "Do you wish to review", a prompt is issued for the SAT keywords previously defined including those that were initially omitted and assigned default values.

The following is a list of the SAT keywords for which prompts are issued when registering a new project with the new_proj command:

Title:
Project title

Investigator:

Investigator Address:

Supervisor:

Address:
Supervisor's name and address. The principal supervisor is the person within the management structure who is responsible for the project.

Supervisor:
Address:
Phone: Supervisor's name and address. The project supervisor is responsible for all aspects of the project. He is the person in direct contact with the daily activities of the project, keeping track of each user's workload and usage requirements. The project supervisor is generally a user of the system and often is the project administrator. A detailed report of the usage of each user on the project is sent to the supervisor by the monthly billing run.

Account:
Requisition:
The account and requisition numbers serve as authorization for the project and ensure that the proper project gets billed for system usage. The rate structure determines the rates at which the project will be charged for its usage.

Amount:
Amount is the dollar limit amount for each project (an unlimited fund or open account can be designated). If the project exceeds the dollar limit, no users on the project are able to log in; however, the projects continues to incur disk and registration charges.

Cutoff date:
The cutoff date is the termination if applicable) for the project. After this date, no users on the project are able to log in; however the project continues to incur disk and registration charges.

```
Is this project delegated?
```

Answer "yes" if the project is to be delegated to a project administrator.

PMF directory:

If a project is going to be delegated to a project administrator, the accounting administrator must be given the Person_id and Project_id of the prospective project administrator and the name of the directory that will contain the project's project master file (PMF).

Enter administrator IDs
(Name.Project).
Type "." to exit

Administrator ID:

If a project is going to be delegated to a project administrator, the accounting administrator must be given the Person_id and the Project_id of the prospective project administrator.

Quota:

Each project is assigned an initial storage system quota. The quota is given in pages; it can be increased or decreased as necessary by the accounting administrator with permission from the system administrator.

Enter initial list of users.
Type "." to exit
Person_id:

Enter the users to be associated with the project; users must have valid person_ids to be listed. At least one user must be registered on the project.

After you have been prompted to supply the information for the above keywords, the new_proj command asks if you want to review the entries you have made. If you answer "yes", the new_proj command displays the information you have supplied and also displays the default values that were entered for the keywords shown below. You now have the opportunity to change any values you choose.

Billing name and address:
The name and address of the individual responsible for billing.

Alias:

An alternate or abbreviated name for the project being registered.

Abs_foreground_cpu:

The value specified in the SAT takes precedence over the Abs_foreground_cpu value specified in the PMF. A value of zero means no limit, so if a value is zero the other one is used, and if both are zero there is no limit.

Authorization:

Requests a range of AIM authorizations at which users on this project may log in. AIM ranges are of the form:

```
low_auth : high_auth
```

A null line specifies that users may only log in at system_low.

Audit
Specifies the audit flags for this project. A null line specifies no access auditing.

Directory quota:
Requests the number of records of directory quota which should be allocated to the project directory. A null line requests the default of 10% of the segment quota.

Project Directory Logical Volume:
Requests the name of a logical volume on which segments subordinate to the project directory will reside. A null line specifies that the project directory will inherit its logical volume from >user_dir_dir.

Project directory access class:
Requests the AIM access class of the project directory. A null line specifies that the project directory will inherit its access class from >user_dir_dir. >user_dir_dir is normally a system_low directory.

Rate structure:
The rate structure determines the rate at which the project will be charged for its usage. For a complete description of rate structures, see the Multics System Administrator's manual.

Group:
The name of the load control group. A load control group is a group of projects that share a guaranteed access to the system.

Groups:
Specifies other load control groups authorized for the project. The Group keyword is used in conjunction with the "igroup" keyword.

Attributes:
Specifies the environment characteristics for the project.

Grace:
Grace specifies the number of minutes the user may be logged in before he becomes subject to preemption. The grace value specified for the project in the SAT, takes precedence over the PMF grace value.

Min_ring:
The minimum ring at which users associated with the project can create a process. The Min_ring value specified for the project in the SAT, takes precedence over the Min_ring value specified in the PMF.

Max_ring:
The maximum ring at which users associated with the project can create a process. The Max_ring value specified for the project in the SAT, takes precedence over the Max_ring value specified in the PMF.

Pdir_quota:
The Pdir_quota value specified for the project in the SAT, takes precedence over the Pdir_quota value specified in the PMF.

Max_background:

The maximum number of concurrent background (absentee) processes that a user associated with the project can have. The Max_background value specified for the project in the SAT, takes precedence over the Max_background value specified in the PMF. A value of zero means no limit.

Max_foreground:

The maximum number of concurrent foreground processes that a user associated with the project can have. The Max_foreground value specified in the SAT takes precedence over the Max_foreground value specified for the project in the PMF. A value of zero means no limit.

EDITING THE PROJECT MASTER FILE

As the second step in the project registration process, you may want to add additional users to the project or change the default keyword assignments initially given to the project's PMF, as shown in the above table. A complete description of all PMF keywords is given at the end of this section in "Project Master File Format". Familiarize yourself with these keywords before attempting to add them to the project.

Also, it is important to note that there are several PMF keywords that have duplicate counterparts in the SAT (see Table 32-3). For these attributes to become effective, they must occur in both files (see "Editing the System Administration Table").

As previously mentioned, a project can be either delegated or undelegated as specified by you at registration time. An undelegated project is not assigned a project administrator at registration time and is under the responsibility of the system administrator. A delegated project is a project that is assigned a project administrator. Each must be edited in a slightly different manner, as described below.

Adding Users and Changing the Default Keywords of an Undelegated Project

After you have registered a project and specified it as "undelegated", you can add users to the undelegated project or change the initial default keyword assignments of the undelegated project's PMF by editing it with the "ec master pmf" command.

The format of the command is:

```
ec master pmf project_name.pmf
```

The Project Master Files for undelegated projects are located in >udd>sa>a>pmf.archive. This command locates the requested pmf in the archive, invokes the qedx editor for you, and returns the PMF to the archive when you are finished. You can then change the PMF according to your specific requirements. Refer to "Project Master File Format" for a detailed description of the PMF keywords.

In the following example, two users are added to the PMF thereby adding two more users who are associated with the project. Note that the information you supply is preceded by an exclamation point (!) for clarity only and is not required as part of the input.

```
!  ec master pmf ALPHA
!  r Alpha
!  /personid:/
!  personid: Brown;
!  a
!  personid: Smith;
!  attributes: multip,vhomedir,nostartup,dailout;
!  personid: Jones;
!  attributes: multip,vhomedir,nostartup,dialout;
!  \f
!  w
!  q
```

For the user attributes to be effective, you must ensure that the attributes are also set in the SAT for the project as a whole. For example, Even though all users associated with the project are not allowed the dialout attribute, the attribute must be set for the project in the SAT for a specific user associated with the project to be granted the attribute in the PMF.

Once you have completed the edit cycle, the pmf is automatically converted to its binary version (the PDT) and the PDT installed in the appropriate place in the system.

Adding Users and Changing the Default Keywords of a Delegated Project

The Project Master File for a delegated project remains under the direct responsibility of the project administrator. The PMF for each project is located within the project directory and can be edited by the project administrator with any text editor available. The PMF must then be converted and installed as described in "Converting and Installing the Project Master File" below.

However, any attributes other than the default attributes the project administrator wants to assign to users must first be granted by the system administrator who will ensure that the appropriate entries are made to the SAT. Then and only then can the project administrator add the attributes to the project's PMF for the user or users.

In the example, the qedx command is used to invoke the editor and add a "limit" statement for user Brown to the PMF named Alpha. The cv_pmf command converts the PMF to a PDT named Alpha.pdt as described below. Alpha.pdt is then installed using the install command, also described below. In this example, an exclamation point (!) indicates lines typed by the project administrator.

```
! qedx
! r Alpha
! /Brown/
  personid:          Brown;
! a
! limit:             800;
! \f
! w
! q
  r 1301 1.202 1.548 96
```

The PMF must now be converted to the PDT, as shown in the example below. A brief description of conversion and installation is given below.

```
! cv_pmf Alpha
  r 1303 1.433  1.321 82
```

The PDT must now be installed, as follows:

```
! install Alpha.pdt
  r 1305 1.702 1.024 31

  From Initializer.SysDaemon (install) 1306.0:
  installed Alpha.pdt for Brown.Alpha.a
```

In the following example, the registration environment characteristics for the project "DOCTEST" are edited using the master.ec in conjunction with the edit_proj command. Changed information that the system administrator typed is preceded by an exclamation point (!) for clarity only and is not required as part of the input.

```
! ec master edit_proj DOCTEST

  Title:                Documentation test project

  Investigator:         M. Bell

  Inv. Address:         CISL

  Supervisor:           Jim Beamer

  Sup. Address:         Andover

  Phone:                492-9310

  Account:              X315

  Requisition:          TB15

  Amount:               open

  Cutoff date:          08/14/85  0000.0 edt Wed

  Billing name:         Michael Bell
```

```
Billing Address:          Cambridge, Ma.

Alias:                    doct

Project administrators. Type "." to delete.
 Administrator:           Bellusci.SysAdmin

Administrator:
 Absentee foreground cpu limit: 3600

 Authorization:           system_low

 Audit:
 Segment quota:           100

 Directory quota:         10

 Rate structure:          default

 Default group:           Other

 Authorized groups. Type "." to delete.
 Group:
 Attributes:              anonymous, multip, bumping, brief,
                          vinitproc, vhomedir, nostartup;

 Grace:                   30

 Minimum login ring:      4

 Maximum login ring:      5

 Maximum pdir quota:      0

 Maximum foreground processes: 1

 Maximum background processes: 1


edit_proj: Do you wish to review the project?   yes
Title:                    Documentation test project

Investigator:             M. Bell

Inv. Address:             CISL

Supervisor:               Jim Beamer

Sup. Address:             Andover

Phone:                   492-9310

Account:                  X315
```

```
Requisition:                TB15

Amount:                     open

Cutoff date:                08/14/85  0000.0 edt Wed

Billing name:               Michael Bell

Billing Address:            Cambridge, Ma.

Alias:                      doct

Project administrators. Type "." to delete.
 Administrator:             Bellusci.SysAdmin

 Administrator:

Absentee foreground cpu limit: 1

Authorization:              system_low

Audit:

Segment quota:              100

Directory quota:            10

Rate structure:             default

Default group:              Other

Authorized groups. Type "." to delete.
 Group:

Attributes:                 anonymous, multip, bumping, brief,
                            vinitproc, vhomedir, nostartup;

Grace:                      30

Minimum login ring:         4

Maximum login ring:         5

Maximum pdir quota:         0

Maximum foreground processes: 1

Maximum background processes: 1


edit_proj: Do you wish to review the project?   no
daily_summary: cut 2, warned 0, total $66156.45
```

As a result of the project registration process, a PMF is automatically created with the default entries shown in Table 32-1. These entries can be subsequently edited. (see "Editing the Project Master File").

Table 32-1. PMF Default Keyword Assignments

Keyword	Default Value
Homedir:	>user_dir_dir>Project_id>Person_id;
Initproc:	process_overseer_;
Attributes:	vinitproc,vhomdir,nostartup disconnect_ok;
Grace:	(project max);
Group:	(project default);
Outer_module:	tty_;
Limit:	open;
Shift_limit:	open,open,open,open,open,open,open;
Cutoff:	open, open, open, never;
Warn_days:	10;
Warn_percent:	10;
Warn_dollars:	10;
User_warn_days:	10;
User_warn_percent:	10;
User_warn_dollars:	10;
Subsystem:	none;
Ring:	(project min), (project max);
Authorization:	"system_low";
Lot_size:	1024;
Kst_size:	1024;
Cls_size:	1024;
Abs_foreground_cpu_limit:	0;
Max_background:	0;
Max_foreground:	0;
Pdir_quota:	0;

CHANGING A PROJECT'S REGISTRATION

The appropriate way to change a project's registration attributes is with the edit_proj command (see the *MulticsAdministration, Maintenance and Operations Commands* manual, Order No. GB64). The edit_proj command provides a single facility for editing all project data.

Setting AIM Attributes for a Project

At project registration time, the new_proj command prompts for the AIM access class of the directory with the project_dir_access_class keyword. A null line specifies that the project directory will inherit its access class from >user_dir_dir. The >user_dir_dir is normally a system_low directory.

Setting Access to a Project's Directory

If a project is delegated, the project administrator should be given status, modify, append access to the directory. All users should given status access to the directory (s *.project_id). The SysDaemon should also be given status, modify, and append permission to the directory (sma *.SysAdmin.*).

Setting Special Attributes for a Project

You can change a projects attributes with the edit_proj command (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64). For a list of attributes see "Keywords" in this section.

UNDERSTANDING THE SYSTEM ADMINISTRATION TABLE

The system administration table (SAT) is a binary table specifying the projects that use the system, the privileges delegated to these projects, and the project administrators. It is located in >sc1. The SAT is maintained by the accounting administrator using the edit_proj command and by the system administrator using the edit_proj and admin_util commands. The system administrator can modify the contents of all SAT entries.

The SAT contains a header and a project entry for each project registered on the system. The SAT header includes the following variables:

- The Person_id.Project_id of one or two system administrators. An asterisk (*) is permitted in either component.
- The user weight table (UWT). This table lists process overseer procedures and an associated weight in load units. (If a process overseer not listed in the SAT is used, a default weight of 1.0 is assigned.)
- Maximum system load units allowed on the system.
- Maximum number of user processes allowed on the system..spb

These variables can be changed using the admin_util command.

Each project entry includes:

- Project_id.
- If the project is delegated, Person_id.Project_id of the project administrators Asterisks are permitted in either component.
- Load control group identifier.
- Alternate load control group identifiers, if any (maximum of two is allowed). An asterisk is permitted. See "Load Control Group" later in this section.
- Maximum bump protection grace time.

- Minimum and maximum execution rings.
- Maximum process directory quota for users on the project.
- Maximum number of processes a user can have on this project.
- Maximum AIM authorization for users on the project.
- Audit flags for users on the project (used with AIM).
- Project attribute flags.

A system administrator may add, delete, and modify project entries by means of the new_proj, delete_proj, admin_util, and edit_proj commands described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

EDITING THE SYSTEM ADMINISTRATION TABLE

The edit_proj command lets you edit the project keyword assignments in the SAT in any of the following ways:

- A prompt can be issued for each keyword in the SAT. The prompts issued are identical to those issued by the new_proj command.
- A single keyword assignment can be changed for a specific project thereby bypassing prompts for keywords that are not to be changed.
- A single keyword assignment can be changed for every project registered on the system.

As demonstrated above, the edit_proj command is a versatile tool for editing the SAT keyword assignments for a project or projects. The description of the edit_proj command below describes how to execute the procedures described in the bulleted list above.

Note the the edit_proj command should be used in conjunction with the master.ec.

The format of the edit_proj command is:

```
{ec master} edit_proj Project_id {keyword {newvalue}}
```

The format of the edit_proj command to change the characteristics of all projects is:

```
edit_proj$change_all keyword {old_value {new_value}} {-long}
```

where:

Project_id
: is the Project_id of the project whose registration data items are to be edited. If this is the only argument specified, edit_proj prints each data item one at a time and waits for a response from the accounting administrator before proceeding. The accounting administrator may respond with any one of the following.

  carriage return
  : to leave the item unchanged

  a new value
  : to replace the printed value

  stop
  : to exit immediately from the edit_proj command without making any changes

keyword
: identifies the item to be changed. The valid keywords are:

```
title
investigator
inv_addr
supervisor
sup_addr
sup_phone
account
requisition
req_amt
cutoff
billing name
billing_addr
alias
administrator
abs-max-fg-cpu
authorization
audit
quota
dir_quota
rate structure
default group
groups
attributes
grace
min ring
max ring
pdir quota
max foreground
max background
```

```
In the new_proj case, this is followed by the

PMF directory name  (if the project is delegated)
project dir access class
project dir logical volume
```

-long, -lg
causes the project name and oldvalue to be printed even when newvalue is given. By default, if newvalue is given the project names and old values are not printed -- only a summary line showing the total number of projects edited is printed.

newvalue
is a new value for the selected field. If newvalue is omitted, the program asks for a new value. If newvalue is given, oldvalue must also be given since these are positional parameters.

oldvalue
is a value for the selected field. Only projects with this value are edited.

Note that the SAT duplicates some of the global keywords found in the PMF for security purposes. The keyword values that you initially assigned in the SAT via new_proj or edit_proj impose a limit on the values in the PMF. Violations of these limits are noted at PMF installation time, with a warning message, but the PMF is installed. If you do not subsequently raise the SAT limits, then at login time the limits are enforced, a warning message is placed in the answering service log, and the user is logged in. (Note, however, that if a user violates a limit imposed by PMF or SAT parameters, by means of a control argument to the login command, then an error message is printed and the user is not logged in.)

SAT keywords entries that have corresponding PMF keywords entries are listed below, including the action taken for each pair of keywords when the PMF and SAT values are in disagreement.

Table 32-2. SAT Keyword and PMF Keyword Correspondence

Global Keyword	Action
abs_foreground_cpu_limit	The smaller of the two values is used as the limit; except that a value of zero means no limit, so if a value is zero the other one is used, and if both are zero there is no limit.
Attributes	Except for the attributes listed below, each attribute is on for a user only if it is on in both the user's PMF entry and the project's SAT entry.
guaranteed_login	On at login time if on in project's SAT entry and on in user's PMF entry and user gives -force control argument in the login command.
anonymous	The anonymous attribute is set only in a project's SAT entry, to give the project permission to have anonymous users. Thus it does not appear in the above list of PMF attributes.
preempting	On at login time if on in project's SAT entry and on in user's PMF entry and user does not give the -no_preempt control argument in the login command.
brief	On at login time if on in project's SAT entry and on in user's PMF entry, or if user gives -brief control argument in login command (permission is not needed to use -brief).

Table 32-3. SAT Keyword and PMF Keyword Correspondence (cont.)

Global Keyword	Action
nostartup	On at login time if on in project's SAT entry and on in user's PMF entry and user gives -no_startup control argument in login command.
no_warning	On at login time if on in project's SAT entry and on in user's PMF entry, or if user gives -no_warning control argument in login command (permission is not needed to use -no_warning)
save_on_disconnect	On at login time if on in project's SAT entry, or if on in user's PMF entry, or if user gives -save_on_disconnect control argument in login command (but note requirement for the disconnect_ok attribute, described above).
authorization	The user's maximum authorization is the lowest of: the value in the project's SAT entry, the value in the user's PMF entry, and the value in the person's PNT entry.
grace	The primary user's grace time is the small of the values in project's SAT entry and the user's PMF entry.
max_background	The smaller of the SAT and PMF values is the user's limit.

Table 32-4. SAT Keyword and PMF Keyword Correspondence (cont.)

Global Keyword	Action
pdir_quota	The smaller of the SAT and PMF values is the process directory quota assigned to the user. If zero in both the SAT and PMF, the pdir_quota in the system default process directory quota (which is specified in the >scl>communications segment) is used.
ring	The user's initial ring is the larger of the initial ring values in the project's SAT entry and the user's PMF entry. The user's maximum ring is the smaller of the maximum ring values in the project's SAT entry and the user's PMF entry.

VIEWING THE CONTENTS OF THE SAT

The entire contents of the System Administration Table can be printed in readable form with the "print_sat project_id" command (see the Administration, Maintenance and Operations Commands manual). This allows you to view all of the data in the SAT and determine if any changes are required. If changes are required, you can then make the changes with the edit_proj command.

SECTION 33

MANAGING USERS

On Multics, the term user applies both to people and logical entities, such as daemons, who have the ability to log in. The ability to log in requires that the user be registered on the system. Each registered user is identified by two items of information:

- Person_id - uniquely identifies the user.
- Project_id - identifies the user as part of a particular resource-control group (project).

If the user process is an absentee process, it is identified by a tag that identifies it as either a foreground absentee process or a background absentee process.

A user can be associated with more than one project. Each combination of Person_id and Project_id identifies a separate user, who is allocated resources according to the identification used at a particular login.

User management on Multics involves the maintenance of the various system tables that contain the information that defines the user environment. The intent of this section is to familiarize you with system tables that contain user registration data and the procedures for maintaining them.

DETERMINING IF A USER IS ALREADY REGISTERED

A table called the Person Name Table (PNT) contains all valid Person_ids registered on the system, in addition to other user values. If you want to determine if a user is already registered on the system, you can print the contents of the PNT with the the print_pnt command. For a description of the print_pnt command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

REGISTERING A USER

Every Multics user must be assigned a valid Person_id and be assigned to a specific project. To register a user, follow the following procedures:

1. Assign the Person_id to the specific project by editing the Project Master File (PMF). This must be done before registering the user. To add the Person_id to the PMF, see Section 32.

2. Register the user with "ec master register". Both commands add entries to the PNT and User Registration File (URF).

   Once these procedures have been completed, the user is registered on the system and is assigned a valid Person_id.

To remove a user from the system, use the remove_user command.

The following is an example of the user registration procedure using the "ec master register" command line.

```
ec master register

Enter full user name (Last, First I.)
Full name           Bell, Michael

Enter mailing address
Address             4 Cambridge Center

Enter programmer number
Prog. number        none

Enter notes
Notes               dummy user

Enter default project
Project ID          MPM
Password:
save
Password again:
save
Network Password:
none
Password again:
none

User ID assigned is "Bell".
Is this OK? yes

More users to add? no
```

UNDERSTANDING THE PERSON NAME TABLE

The Person Name Table (PNT) is a multisegment file referenced by the system when a user attempts to log in: it contains all valid Person_ids. Associated with each Person_id in the table is a login alias, user flags, and encrypted password information. The registration process creates an entry in the PNT for the newly registered user.

Note that, while portions of the user entry in the PNT are stored in encrypted form, any encryption algorithm is susceptible to a sophisticated, computer-assisted code-breaking effort. Therefore you should ensure that access to the PNT is as restricted as possible. In general, only the SysAdmin and SysDaemon projects should have access to the PNT.

To manipulate the PNT, the following privileged gates are available:

- pnt_login_gate_ - for identification and authentication upon login. The only userid which needs access to this gate is Initializer.SysDaemon.z.

- pnt_admin_gate_ - for System Administrative functions. All people who are allowed to read or write PNT entries need access to this gate. No passwords can be retrieved using the entries in this gate. The recommended access to this gate is re for Initializer.SysDaemon.z and re .SysAdmin.

- pnt_network_gate_ - for IMFT, Card_Input, and IO.SysDaemon manipulations of "network" (formerly "card input") passwords. All userids which manipulate and validate network passwords need re access to this gate. This includes the IMFT daemons, and any daemons which manipulate card input.

- pnt_priv_gate_ - allows access to passwords. Not used at this time, but available for very privileged applications which must perform statistical analysis of passwords. Access to this gate should not be given except to the most trusted of system administrators, and only if absolutely necessary. There is no need to have any access on this gate.

- pnt_fs_gate_ - This is not a privileged gate. The pnt_fs_gate_ gate allows one to list and set the acl on the PNT providing you have the appropriate access to the containing directory (usually >scl). Access should be set for re ...

The following is a sample PNT produced with the print_pnt command:

```
User ID:      Swenson
Full name:    Swenson, E
Address:      CISL
Notes:        Hardcore Unit
Project ID:   Multics
Alias:        ejs
Flags:        password,^network_pw,^trap,^lock,change,
                ^must_change,^generate,operator,^time_lock
Authorization Range:  system_low:system_high
Audit Flags:  fsobj=N/R,fsattr=N/R,rcp=N/R,admin=N/N,special=N/M,
                other=N/R,admin_op,priv_op,fault,^small_cc,^moderate_cc

   1185 good passwords given, last at 01/10/85  1228.7.
   32 bad passwords given, last at 01/08/85 1650.8 from ASCII VIP7801
   terminal none.
```

UNDERSTANDING THE MAIL TABLE

The Mail Table (MT) is a table of names and corresponding mailing addresses. All users are registered in the mail table. It enables users to address one another by name or alias, without specifying the addressee's Project_id. For instance, a user could send mail to "Rickert" without knowing whether Rickert is on the Mktg or Sales project. The mail table can also include names for mailing lists and Forum meetings.

Each Multics system has one mail table. The system administrator creates the initial mail table with a create_mail_table command. Mailing addresses consist of the Person_id and the default project for each user registered in the PNT. As new users are registered, a corresponding entry is automatically added to

the mail table. Similarly, if a user changes his default project at login time, his mail table entry is also updated.

The mail table entries are case-insensitive, that is "SMITH", "Smith" and "smith" are known as the same Person_id. However, the PNT is case-sensitive so that "SMITH", "Smith" and "smith" are three different names. Since the mail table and the PNT contain a parallel set of entries, it is important when choosing new Person_ids to consider possible name conflicts in a case-insensitive manner.

UNDERSTANDING THE USER REGISTRATION FILE

The User Registration File (URF) is a multisegment file containing each user's full name, mailing address, programmer number, log-in alias, and Person_id. This data base is referenced when registering a new user and when performing administrative operations such as printing mailing labels.

You can print the contents of the URF by using the print_urf command. For a full description of the print_urf command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

MODIFYING THE PERSON NAME TABLE AND THE USER REGISTRATION FILE

Once a user has been registered on the system with "ec master register" (or the new_user command), entries for that user are stored in the PNT and the URF. If, for any reason, you want to change the entries in the PNT or URF for the user, you can do it with "ec master change". For a full description of the change entry point to the master.ec, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

The "ec master change" command entry point lets you change the following for a specified user:

- Full user name
- User mailing address
- Programmer number
- Default project
- Login password
- Card input password

Note that you cannot change the registered Person_id of the user. If you make an error typing in the user's surname that is to be used as the Person_id, you must re-register the user and delete the misspelled Person_id.

There are several variations of "ec master change" that let you change some of the individual items in the PNT and avoid the need to type the item name or to be prompted for new values for all items. The variations of "ec master change" are: chaddr, chalias, chpass, chcpass, chdf_proj, chname, and chprog. Each of these commands must be used with the master.ec.

In the following example, "ec master change" is used to change the "address" for user Smith. Note how "ec master change" displays the current information and then prompts for the new information. If you do not want to change a specific field, simply hit the RETURN key. In the example below, user input is preceded by an exclamation point (!) for clarity only.

```
! ec master change Smith

   Full name:      Smith, John
   Full name
   Address:        Cambridge, Massachusetts
   Address         ! Billerica, Massachusetts
   Prog. number:   none
   Prog. number
   Notes:
   Notes
   Project ID:     SysAdmin
   Project ID
   Password
   Network Password
```

The new_user command also lets you modify values in the URF and the PNT. However, only members of the SysAdmin project have access to the new_user command. In addition to changing the same items as "ec master change", new_user lets you change user alias, password flags, AIM authorization, default AIM attributes, and audit flags.

MODIFYING THE MAIL TABLE

When a person is registered, the procedure described under "Understanding the Mail Table" is used to form his mailing address; however, a person can change his own mailing address using the set_mailing_address to anything he wants. The user can also display other mailing addresses with the display_mailing_address command.

Mail table entries which do not correspond to registered users have several uses. They might be used for automatic forwarding of mail to other computers on a network. They may also allow mail to be sent to a mailing list or Forum meeting without requiring that the sender know that the recipient is not a real user. This allows incoming mail from other computers to be addressed to such objects, even if their mail software makes it difficult to use Multics' syntax for such addresses. A system Administrator can add arbitrary entries to the mail table, and can delegate maintenance of such entries to a specified user or users by means of ACS segments.

The mail table is implemented as a multisegment table (MSF managed by ms_table_mgr_), just as the PNT and URF currently are. Its pathname is

>site>mail_system>mail_table

I resides in ring-2 so that it is protected from improper updates. Two types of entries exist in the data base: regular entries and alias entries. Recorded in each regular entry are the following: the default project, a flag specifying whether the entry corresponds to a PNT entry, the ACS pathname, the mailing address, and the first and last aliases (the head and tail of a doubly-linked list of alias entries). Alias entries will contain only the primary name, a flag specifying whether the alias corresponds to a PNT alias, and the previous and next aliases. In the mail_table_entry structure, the flags.alias_entry component is used to distinguish the two types of entries. In most cases the interfaces deal only with regular entries; the exception is that all the aliases of an entry are deleted when the entry itself is deleted.

Once the mail table has been created, the system administrator can add to it with the add_mail_table_entry command. This command adds an entry to the mail table and specifies the entry's mailing address. The name must not already exist in the mail table or the person name table (PNT). The system administrator can modify an entry in the mail table by using the update_mail_table_entry command. The delete_mail_table_entry command is used to delete names of users who are no longer registered and of mailing lists and Forum meetings that no longer exist.

A person name table (PNT) alias can be deleted (via ec master chalias), but the associated mail table entry and mail table alias are not deleted. This is because a user may be accustomed to sending mail by using an alias only. If the mail table alias was deleted when the PNT alias was deleted, the mail could possibly be sent to the wrong user. However, the system administrator can manually delete the alias by using the delete_mail_table_entry command.

Users are able to modify the mailing address information in their own entry. If there is a non-blank acs_path component, any user with rw to the segment specified by the pathname may update the mailing address information in an entry. When a user's default project is changed (either by logging in with the -change_default_project control argument or by a system administrator using new_user$cg), the default_project in the mail table will be updated. In addition, new_user will create an entry for a user when he is registered on the system, and create an alias entry if the user is given an alias. The remove_user command will delete the appropriate entries from the mail table. The add_mail_table_entry, update_mail_table_entry, and delete_mail_table_entry commands may also be used by administrators to maintain entries that are not associated with registered users or to manipulate entries in ways inwhich normal users are prohibited. For a description of the mail_table_entry commands, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Three gate procedures are provided to access the mail table.

1. Normal users will use the mail_table_ gate to retrieve and update the information in the mail table. In general, it is called by address parsing routines in mail_system_ and by the command provided for users to update their mailing address.

2. The Answering Service will use the mail_table_initializer_ gate to update an entry when a user changes his default project while logging in.

3. System administrators will use the mail_table_priv_ gate to create and manage the mail table. It is called by existing system administrative commands (primarily new_user) and by commands provided with the mail table software.

MODIFYING AIM AUTHORIZATION FOR A USER

You, the system administrator only, can change the AIM authorization for a user by using the new_user$cga command. For a description of this command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

MODIFYING USER AUDIT FLAGS

Audit selectivity flags can be changed by using either new_user$cga for individual users, or the set_sat_audit command for projects. For details on either of these commands, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64. For detailed information on audit selectivity, see Section 25.

ADDING AN ANONYMOUS USER TO AN UNDELEGATED PROJECT

Some projects need to allow users not registered on the system to log in under their project. This ability is provided by Multics through the "anonymous user" mechanism. Anonymous users are not distinguished by the system for the purposes of access control (which is the reason for the requirement that persons be registered) or billing. An anonymous user account does not require a password but one can be assigned.

Strict controls should be exercised in the establishment of an anonymous user account via a special process overseer, limited subsystem, or access controls.

To add an anonymous user to the system, follow the following steps:

1. Register the user with "ec master register" and assign a Project_id of "anonymous".

2. Edit the Project Master File (see Section 32) of appropriate project and add the user keyword:

```
personid: ;
```

This is the only statement required in the PMF for an anonymous user.

Note that all the user keyword statements are valid for anonymous users and can be specified after the personid statement. Global default values specified for the project apply to the the anonymous user whenever a user local keyword statement is omitted (see "Project Registration").

Anonymous users log in to Multics with either the enter command or enterp command depending on whether or not the password statement is given after the anonymous user entry in the PMF.

The enter command is used to log in an anonymous user without a password; the enterp command is used to login an anonymous user with a password. An anonymous user has a process created for him when he logs in. Depending on the environment characteristics set for the anonymous user, access to all Multics facilities can be set (if his process overseer procedure is process_overseer_), or some specialized subsystem (e.g., the Multics FAST subsystem or even a user-written environment specified by a pathname).

Anonymous user system usage is aggregated on the monthly bill and the statistics under the heading "Anonymous Users." The who command lists them as "anonymous"; the as_who command lists them with an asterisk in front of the name given with either the enter or enterp commands. There is no reason why an anonymous user cannot be a project administrator.

A sample anonymous user entry in a PMF is shown in the example below:

```
personid:       *;
password:       anon;
initproc:       >udd>Project_id>login_proc;
homedir:        >udd>Project_id>users;
```

Since the password statement is given, all anonymous users who log in under this project must do so via the enterp command and give the "anon" password. These users are subject to a special process overseer. Also, they all share permanent storage in the specified home directory. Global keyword default values apply for the remaining unspecified user keywords.

SPECIAL USER IDENTITIES

Several special user identities are built into the organization of the Multics system, or are referred to in Multics documentation by name. This section describes the various special identities and their characteristics.

The special user identities that Multics requires are created automatically when a new Multics site is initialized. However, some special user identities might require some "fine tuning" specific to your Multics site. You can change the environment characteristics of special user identities the same way you change the environment characteristics for any user (see Section 32).

The following projects and special user identities are automatically registered and installed during initialization of a new site:

- SysAdmin
  - SA1.SysAdmin
  - Repair.SysAdmin
- SysDaemon
  - Initializer.SysDaemon
  - IO.SysDaemon
  - Backup.SysDaemon
  - Dumper.SysDaemon
  - Retriever.SysDaemon
  - Ring_1_Repair.SysDaemon

- Repair.SysDaemon
- Salvager.SysDaemon
- Scavenger.SysDaemon
- Utility.SysDaemon

- Daemon
  - Card_Input.Daemon
  - Metering.Daemon
  - Volume_Dumper.Daemon
  - Volume_Retriever.Daemon
  - Volume_Reloader.Daemon
  - Data_Management.Daemon

The following projects and identities are registered in the SAT when system is initiated for the first time, but the PDTs are not installed. Users on these projects cannot log in until their PDTs are installed by a system administrator.

- HFED
  - anonymous.HFED

- Terminals
  - anonymous.Terminals

- Operator
  - Operator.Operator

SPECIAL SYSADMIN USER IDENTITIES

The special SysAdmin user identities are Repair.SysAdmin and SA1.SysAdmin. SysAdmin processes are, by default, given access to major directories and potential access to all segments and directories in the hierarchy.

The Repair.SysAdmin identity is provided for emergency fixes by system programmers who need SysAdmin access privileges. The password for this user should be kept in a sealed envelope in the operations area. (Since the Person_id Repair is also used for Repair.SysDaemon, the password for these two users is the same.)

The Repair.SysAdmin identity is also useful when a new site is bringing up the system for the first time. Besides the SysAdmin access privileges, it has the normal initial procedure, process_overseer_. (The only other user on the SysAdmin project at this time, SA1, has accounts_overseer_ as its initial procedure; i.e., SA1 is an accounting administrator.)

SPECIAL SYSDAEMON USER IDENTITIES

The special SysDaemon user identities are, by default, given access to all segments and directories in the hierarchy. This is necessary so that the various daemon processes can perform their functions. Due to the broad access capabilities enjoyed by these processes, strict control must be exercised over their use. Many sites will find it desirable to insist that some of the daemon processes defined below only be logged in via the message coordinator. This has the advantage that the passwords for such daemons need never be disclosed by the system administrator. Furthermore, all use of these daemons can then be logged by the message coordinator.

Initializer

Initializer.SysDaemon is the User_id assigned to the first process created during system initialization. This process remains in existence all the time the system is operational. It runs several subsystems critical to system operation, including the answering service (for login processing and process creation), administrative table installations, the message coordinator, and system control functions (e.g., hardware reconfiguration).

This process communicates with operators and administrators via the message coordinator, usually to a console physically located in the computer room.

The User_id of this process is not listed by the who command (described in the *Multics Commands and Active Functions* manual, Order No. AG92), since it is always present whenever the system is operational.

I/O

IO.SysDaemon has control of the local and remote printers and punches, and the remote card input, on the system.

Incremental Backup

Backup.SysDaemon operates in ring 1 instead of ring 4, so that it can dump all segments that are part of the hierarchy. It comes up in ring 1 because the statement:

```
ring:      1, 1, 1;
```

has been added to its PMF entry and because the SAT gives permission to the SysDaemon project to have users log in in lower rings. The initial procedure for the Backup.SysDaemon is process_overseer_.

Complete Dump

Dumper.SysDaemon is used to perform complete dumps of the hierarchy and operates just like Backup.SysDaemon. Dumper.SysDaemon operates in ring 1.

Retriever

Retriever.SysDaemon is used to retrieve user segments from complete, consolidated, or incremental hierarchy dump tapes. It operates in ring 1, and its initial procedure is process_overseer_.

Ring 1 Repair

Ring_1_Repair.SysDaemon is used to fix problems with access control in ring 1. This identity uses the normal initial procedure, process_overseer_, but in ring 1. It is one of the users who can execute any command in ring 1. The password for this user should be kept in a sealed envelope in the operations area.

Repair

Repair.SysDaemon is used for emergency fixes by system programmers who need SysDaemon access privileges. This identity uses the normal initial procedure, process_overseer_, in ring 4. The password for this user should be kept in a sealed envelope in the operations area.

Salvager

Salvager.SysDaemon is a daemon used to perform directory and quota salvaging. It has a completely different function from that of Scavenger.SysDaemon. It is used to correct quota inconsistencies after ESD-less crashes (or at any other time when discrepancies are detected), correct inconsistencies in the internal structure of directories, and compact directories to recover wasted space. Salvager.SysDaemon is usually invoked via the admin.ec "x repair" command which logs in the daemon and determines it mode of operation. The start_up.ec >t>salvager_start_up.ec, provides the script of commands executed by the Salvager.

Scavenger

Scavenger.SysDaemon is a daemon used to perform volume scavenging. Volume scavenging is similar to volume salvaging, but can be performed when the physical volume in question is mounted and in-use. The daemon is usually invoked via the admin.ec "x scav" command which logs in the daemon, and gets the scavenge going. The start_up.ec >t>scavenger_start_up.ec, provides the script of commands executed by the Scavenger.

Utility

Utility.SysDaemon is a daemon used to perform a variety of utility functions. It is logged in at system start up time and remains logged in during the bootload. It's start_up.ec (>t>utility_start_up.ec) directs its operations. The standard version of this start_up.ec causes Utility to copy crash dumps from the DUMP partition into the hierarchy, delete old process directories, poll the MPCs, check for memory errors, and monitor quota in critical system directories.

SPECIAL DAEMON USER IDENTITIES

The Daemon project is given no special access. It is used to allow unprivileged daemon processes to be logged in by the operator via the message coordinator so the operator can perform system functions that do not require privileged access. Examples of these functions include reading cards, printing dumps from previous crashes, and metering system performance. Each of the special user daemon identities are described below.

Card-Reading Daemon

Card_Input.Daemon is used to read card decks that have been submitted to operations by users. The operator command:

```
exec read_cards
```

logs in the daemon and starts reading cards.

Metering Daemon

Metering.Daemon is used to make periodic measurements of system performance and make the results available to the system administrator. To use this daemon, the site must provide one or more exec_com segments and place them in the >udd>Daemon>Metering directory. The start_up.ec segment for Metering.Daemon must initialize the taking of periodic measurements by using one of the system timed-event managers (e.g., the memo command with the -alarm, -repeat, and -call control arguments). The system administrator should work with the site analyst to decide what measurements to take and what metering commands to use.

Volume Dumper Daemon

Volume_Dumper.Daemon is used to perform incremental, consolidated, and complete volume dumps. It operates in ring 4, and requires access to these gates: hc_backup_, hcs_, rcp_sys_, mdc_, and rcp_.

Volume Reloader Daemon

Volume_Reloader.Daemon is used to recreate the logical image of a failed physical volume. It operates in ring 4, and must be run from the same working directory, with the same access requirements, as that of Volume_Dumper.Daemon.

Volume Retriever Daemon

Volume_Retriever.Daemon is used to recover recently damaged segments or subtrees. It operates in ring 4, and requires the same access and the same working directory as that of Volume_Dumper.Daemon.

Data_Management Daemon

Data_Management.Daemon must be logged in to run Data Management on Multics. It serves as caretaker during DMS operations by recovering abandoned transactions and transactions belonging to dead processes. It starts DMS initialization after a Multics bootload and performs crash recovery. It is also the DM daemon that schedules and controls Data Management shutdown.

Data_Management.Daemon has its own process overseer (dmsd_overseer_), which handles all conditions and signals to the daemon, sets up an event channel for requests to the daemon, and registers itself with the answering service to be notified of process terminations.

The DM daemon requires re access to the dm_hphcs_, dm_admin_gate_, and dm_daemon_gate gates.

SPECIAL HFED, REPAIR, AND OPERATOR USER IDENTITIES

Personnel that require access to the system for maintenance, repair, or system operation are assigned distinct projects and anonymous user identities. This allows any individuals working in these areas to gain access to the system.

Field Engineering Personnel

Field engineering personnel, who need access to the system for maintenance purposes, are given the HFED project with one anonymous user. The site may choose to alter this initial registration to meet its own requirements.

Field engineering personnel should receive a copy of all system maintenance reports generated at the site.

If the Total Online Testing System (TOLTS) or the Peripheral Online Testing System (POLTS) is to be used on the system, a special project (such as TOLTS) should be set up for the use of field engineering personnel. A special initial procedure, tolts_overseer_, is provided for such use. Field engineers who have tolts_overseer_ as their initial procedure are restricted and checked in their actions; field engineers with the normal initial procedure, process_overseer_, are not. The system_start_up.ec segment must also contain this command:

```
hpsa >system_library_1>phcs_ re *.TOLTS.*
```

to allow the TOLTS project the access required for its operation.

Terminal Repair

The Terminals project is for remote-terminal service personnel who need to log in and check a terminal. This project is set up with one anonymous user whose initial procedure is terminals_overseer_. (If other users are added to this project, they should also have the terminals_overseer_ initial procedure.) The preempting attribute should be given to users on this project and the grace should be set at only 5 minutes. These restrictions mean that service personnel can log in and use the system long enough to check a terminal without bumping other users (e.g., system programmers).

The terminals_overseer_ initial procedure locks the user into a closed subsystem that restricts him to a special set of commands. It is recommended that the site make copies of this appendix available to remote-terminal service personnel.

One of the commands, test, actually prints test.runoff in the user's working directory. This segment does not exist unless created by the system administrator. It is up to the system administrator to write a test.runoff segment that can adequately test the kind of terminals used at the site.

System Operators

The Operator project is for all the system operators. An administrator should register each operator by name on this project. Since the operators are given access to the system through this separate project, they will not need to log in as one of the daemons for their own personal use or for experimenting with the system.

Some sites find it convenient to allow users to send interactive messages to the operator. One way of doing this is to have an operator logged in from a normal terminal in the machine room, using a fictitious identity (e.g., Opr.Operator) and inform all users of this identity.

Another way is to have a daemon accept messages for the operator and print them on a message coordinator terminal.

CREATING A FICTITIOUS USER

Occasionally, an administrator may want to register a full user identity that corresponds to no real person. Although this can be done, it should be discouraged unless a good reason can be advanced, since it tends to circumvent the access control mechanisms. To add a fictitious person:

1. Enter "ec master register"

2. Enter the name of the person in charge of the identity preceded by an asterisk (*). The asterisk before the name prevents the system from attempting to generate a person_id from the name; instead, the system asks for a Person_id.

3. Enter the fictitious Person_id in response to the prompt.

4. Once the fictitious user is registered, enter the person_id in the appropriate PMF.

SYSTEM PROGRAMMERS

If significant system programming work will be done on the system, it is best to group the system programmers on one or two projects. Users on these projects can then be given access to the privileged gate phcs_ to investigate the contents of the supervisor. Since they have privileged access to the system, only responsible system programmers should be registered on these projects.

To give these projects access to the phcs_ gate, the following line must be added to the system_start_up.ec segment:

```
hpsa >system_library_1>phcs_ re *.Project_id1_.* re *.Project_id2_.*
```

Some sites omit this line, for additional security, and set up a section of admin.ec that gives access to phcs_ in response to an operator command. Such access goes away at the next bootload. Without access to phcs_, a system programmer may be unable to diagnose a hardware or software problem, since he is unable to examine any ring 0 data except that listed in ring_zero_meter_limits_ASCII_.

TAILORING THE USER ENVIRONMENT

You can assign users a great variety of system features and attributes that create a specific environment for the user or group of users. You do this by:

1. Editing the values in the user's Project Master File.

2. Editing the values in the System Administration Table, if required.

3. Changing the values in the URF and PNT if you want to alter the user's registration characteristics.

For a complete description of how to change the environment characteristics of a user, see "Project Registration" in Section 32.

PART IX
CONTROLLING SYSTEM USAGE

SECTION 34

MANAGING SHIFTS

System usage may be divided into a maximum of eight shifts, numbered from 0 to 7. These shifts may begin at any half-hour during a week. Each shift has a number assigned to it. For example, Shift 1 may indicate Sunday 2400 to 0800. Shift numbers always remain the same unless they are changed in the shift table.

SHIFT TABLE

Shifts times are set by filling a table that contains 336 entries (one for each half-hour in the week) with a digit from 0 to 7. The system administrator must specify the shift_table parameter with the ed_installation_parms command (described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64) to inspect or modify this table. The default shifts 1,2,3, and 4 are:

```
shift 1 0800-1800 weekdays
shift 1 1800-2400 weekdays
shift 3 0000-0800 every day
shift 4 0800-2400 weekends
```

Use the ed_installation_parms command as follows:

```
ed_installation_parms
c shift Fri 1800 thru Mon 0800 5
w
q
```

This defines shift 5 to start at 1800 Friday and end at 0800 Monday.

USING THE SHIFT CHANGE EXEC_COM

The shift_config_change.ec is an executive command that contains commands to alter system parameters or take any other action required by a particular site.

As the system administrator, you are responsible for writing an exec_com according to the needs of your site. For example, if you did not want to run any absentee jobs during shift 1, you could write the shift_config_change.ec to shut off the absentee facility during shift 1 and reactivate it at the next shift. The shift_config_change.ec segment is kept in the >sc1 directory. It is executed at startup time, at each shift change, and each time the operator types the maxu auto (1) or maxu level (2) command described below.

The exec_com is called as follows:

```
ec shift_config_change oshf nshf auto_maxu ncpu nkmem
```

where:

- oshf
  is the old shift

- nshf
  is the new shift

- auto_maxu
  is a three-valued switch, where:

  "0" means that load limits have been set by the operator and are not being adjusted automatically by load control.

  "1" means that automatic load limit setting (as a function of the shift and configuration as specified in the configuration array) is in effect.

  "2" means that load leveling is in effect. Accounting control, which runs every 15 minutes, attempts to adjust load units automatically. This switch should be used sparingly since it is not accurate.

- ncpu
  is the number of CPUs in the configuration.

- nkmem
  is 1000 words of main memory in the configuration.

The shift change exec_com can be used to set parameters that are not specifiable in the configuration array in installation_parms. (You can also use it to replace the setting of parameters from the configuration array.) Normally, the load limits are automatically adjusted after the shift_config_change.ec is called. However, if auto_maxu equals 0 (this means that the load limits are not adjusted automatically by load control) you can use the shift_config_change.ec to set the load limits.

The shift_config_change.ec must be written so that it can deal with recursive execution in case the exec_com executes either the shift command or the auto_maxu command with auto (1) or level (2) arguments.

SECTION 35

MANAGING SYSTEM LOAD/ ALLOCATING PROCESSOR RESOURCES

A load control group consists of one or more projects that share guaranteed access to the system. Each load control group has a quota of *primary* load units. This represents a guaranteed number of users within the group who can always login. If the group's primary quota is full and you attempt to log in to the same load control group, the system assigns you *secondary* status. If both the group's primary quota and the system become full, only primary users can log in by preempting (bumping) secondary users.

CONTROLLING LOAD CONTROL GROUPS

The two types of administrators who control load control groups are system and project administrators. A *system administrator* assigns each project to a load control group in the System Administrator Table (SAT).

Project administrators can control certain attributes to allocate resources among users assigned to particular projects. The control is subject to the values that the System Administrator has put into the SAT table for that particular project. The project administrator can control the following attributes:

- primary status
- secondary status
- preemption
- grace time

These concepts are explained in the sections below.

Primary and Secondary Attributes

The project administrator controls who has *primary* and *secondary* status by adding specific attribute statements in the Project Definition Table (PDT). (See the discussion on Managing Projects in this manual.) Secondary users can log in when the system is not full, but when the system becomes full they can be preempted (bumped) by a primary user who wants to log in. The system administrator can set specific attributes for each project and the project administrator can set specific attributes for individual users on his project, subject to the limits set by the system administrator.

The following attribute statement provides secondary status only:

```
attributes:     no_primary;
```

This statement allows users to utilize the system when it is not full, but the system may preempt these same users whenever the system starts to become full. The no_primary attribute also limits users from competing with other users of the group for primary status.

The following attribute statement provides primary status only:

```
attributes:     no_secondary;
```

This statement allows users to log in when the system is full provided there is a secondary user on the system who can be preempted or there is a primary user whose grace period has expired. This statement only allows a user to log in if he will not exceed the maximum number of users for his load control group.

You should never assign both the no_primary and no_secondary attributes to a user. When both attributes are found in the project master file (PMF) the cv_pmf prints a warning message even though the PMF is still converted.

You can use the following attribute statement for system maintenance projects:

```
attributes:     guaranteed_login
```

If this attribute is in a user's PDT and the project has the same attribute in his SAT, he can log in (if at all possible) when he specifies the -force control argument to the login command. He can exceed the load control limits of the system in the process of logging in.

Preempting and Grace

The project administrator controls preemption by adjusting two PMF parameters: the *preempting attribute* and the *grace time*.

Preemption within a load control group prevents primary users from monopolizing the system. Although secondary users generally utilize the system when it its not full, they can preempt primary users in their group if the primary user's grace time has expired. The following attribute statement allows secondary users to preempt primary users in their group (after the grace time has expired):

```
attributes:   preempting;
```

To preempt another user, a user must have the preempting attribute in his PMF and his project must have the preempting attribute in the SAT. When the system preempts a user, it locates the user who logged in first from the same load control group as the user who is doing the preempting.

If a secondary user does not have the no_primary attribute, the system can promote this secondary user to primary status when a primary user logs out. To do this, the system locates the load control group of the primary user who logged out last and promotes the secondary user who logged in first from the same load control group.

Grace is the amount of time in minutes the primary user may be logged in before he becomes subject to preemption. This time period controls how long users may monopolize primary status. In order to be preempted, a primary user's grace time must be expired. After the grace period is exceeded a primary user becomes a secondary user. The project administrator sets the grace time for a particular user with the following user grace statement:

```
grace:      N
```

N is the number of minutes the user may be logged in before he becomes subject to preemption.

The project administrator sets a default value for all users of a project with the following global grace statement:

```
grace:      N
```

N is the number of minutes each user may be logged in before they become subject to preemption.

The system administrator sets the maximum grace time for the project in the SAT. The system uses the maximum grace time if the project administrator does not specify either a global or user grace statement. If the two differ, the smaller value applies.

DEFINING LOAD CONTROL GROUPS

To define load control groups, use the accounting start up exec_com command. This command sets up the accounting system with the following load control groups:

- System - System administration personnel and daemons use this group.
- SysProg - System programmers use this group.
- Other - All other systems users are placed in this group. The new_proj command automatically assigns new projects to this group.

To define additional load control groups, the system administrator must use the ed_mgt command to edit the Master Group Table (MGT). See the discussion on "Master Group Table" later in this Section.

Individual Load Control Groups

Normally, the SAT entry for a user's project determines his load control group and all users on a project are in the same group. However, the system administrator can place individual users in a private work class indirectly. To specify the group for each user on that project in the PMF entry for that user, the system administrator should perform the following functions:

- Give a project the "igroup" (individual group) attribute.

The system administrator can use the "groups" keyword to the edit_proj command to set a project's authorized groups (see Section 32). The project administrator (who may be the system administrator) can perform the following functions:

- In the project PDT, gives the "igroup" attribute to each user who is going to be in a different load control group than the project's default group.
- Enters the name of the private load control group in the user's PDT entry.

For a complete description of Project Registration, see Section 32.

To define the group and work class used by the individual user and to assign the group to the work class, the system administrator can use the ed_mgt command, described in the *Multics System Maintenance Procedures* manual, Order No. AM81.

Setting Up Load Control Groups

To set up a load control group, you first must determine the required groups and which projects belong in each group. Next decide how much of the system maximum load units each group can occupy. These decisions require a picture of the normal load pattern of the system, as well as knowledge of the management's priorities attached to individual projects. Beginning sites should stay with the three default groups (System, SysProg, Other) until they have built up the required experience.

Load Control Equations and Group Parameters

An explanation of how to calculate values for some of the load control parameters in the Master Group Table (MGT) is provided here. The system calculates values based on the numbers that you supply. These calculated values serve to establish the number of users that can log on in different situations.

The following equations actually determine the maximum load units (rather than the maximum users) for the group. Each user receives a load control weight of one unit (by default), making the number of users from each group equal to the number of load units. You can also assign weights other than one.

The max_prim load control equation for a group determines the maximum number of users in the group that can log in as primary users. Additional users from the group can log in as secondary users (subject to preemption) if the system is not full and the group's absolute_max figure is not exceeded. The max_prim load control equation is:

```
max_prim = minu + (num/denom) * system_maxu
```

max_prim
: This number represents the maximum number of users that can log in as primary users.

minu
: This number represents the minimum number of primary users allowed to log in.

denom
: This number represents a percentage of the system_maxu figure. You can set this number to the typical system_maxu figures (see definition below).

num
: This number represents the desired number of users from the group (less the minu_figure) that should be given primary status on a full configuration.

system_maxu
: This number represents the maximum units for the system less the number of consoleless daemons and absentee jobs currently logged in.

The following example for the max_prim equation shows a system with an 80-user capacity that normally runs 7 daemons and 3 absentee users. If you want this group to have 15 primary users, with a minimum of 3, set:

```
minu =  3
num =   12
denom = 70
system_maxu = 70  (80-7-3)
```

The equation would be filled in as follows:

```
3+(12/70)*70 = 15
```

The absolute_max equation determines the absolute maximum number of users from the group (both primary and secondary) that can log in. This limit is not exceeded even if the system is not full. If you do not want to impose an absolute_max limit on the group, set the value of "3276.7" for the minamax parameter of a group. The absolute_max equation is:

```
absolute_max = minamax + (numl/denoml) * system_maxu
```

absolute_max
: This number represents the absolute maximum number of primary and secondary users from the group that can log in.

minamax
: This number represents the number of users that a load control group can have logged in in a minimum configuration.

denom 1
: This number represents a percentage of the system_maxu figures. You can set this number to the typical system system_maxu figures.

num 1
: This number represents the desired number of users from the group (less the minu_figure) that should be given primary status on a full configuration.

system_maxu
: This number represents the maximum units for the system less the number of consoleless daemons and absentee jobs currently logged in.

The following example for the absolute_max equation limits this group to 30 users with a full configuration, but to no fewer than 15 users on any configuration. To accomplish this set:

```
minamax   = 15
numl      = 15
denoml    = 70  (80-7-3)
```

The resulting equation would be:

```
30 = 15+(15/70)*70
```

Setting the num or numl to zero disables the proportional load unit for the group, leaving the respective units figures as a constant, not dependent on system_maxu.

The following list gives the keywords used in the ed_mgt change request to change the values of the variables in the load control equations:

Variable Keywords	Alternate Form
max_prim	max_prim, maxp, maxu, m
minu	constant, const, con
num	num, numerator
denom	denom, den, denominator
minamax	minamax, abs_max, abs
numl	numl
denoml	denoml, denl

The variable shown as "minu" in the equation is the only one whose name is not one of the keywords accepted by the change request to change that variable. (Three variations on the word "constant" are used instead.)

MASTER GROUP TABLE

The master group table (MGT) is a segment containing information work class and load control group membership of each user is determined from information in the SAT and PMF. The system administrator maintains and edits the MGT using the ed_mgt command which is described in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64. Following the description of the ed_mgt command, you can find examples that illustrate both the use of the command and the use of the load control groups and work class.

Working through the examples on a temporary copy of the MGT (using the "p*" request of the ed_mgt command to display the contents of the MGT after each operation) helps you to visualize the relationships between load control groups and work classes and to understand their use.

SECTION 36

MANAGING ABSENTEE USAGE

There are two types of absentee processes that can be created by users:

- Foreground absentee processes
- Background absentee processes

A foreground absentee process is created in response to either a user's login request or a request in the foreground absentee queue: the request is immediately executed. A background absentee process is created in response to a request in one of the background absentee queues (queues 1 through 4); background absentee requests are executed at a later time depending upon the time specified and system load control.

Management of absentee usage involves determining the projects and users that can enter absentee requests as well as the number of absentee requests that can be entered on behalf of each user or project. You can also specify the amount of cpu time to be available for the execution of foreground absentee processes.

UNDERSTANDING HOW ABSENTEE USAGE DIFFERS FROM INTERACTIVE USAGE

Multics permits users to execute programs or run jobs in real time, or execute programs or run jobs at a later time. Real time execution permits users to execute programs as soon as they issue the request; absentee requests permits delayed execution of the job or program until a time specified by the user. Interactive usage is anything the user does while logged in; absentee usage is the execution of commands, programs, or jobs performed by a special absentee process created by the user. The absentee process executes whatever the user has specified at a time specified by the user.

The principal difference between an absentee process and an interactive process is:

- The I/O switch user_input is attached to a control segment containing commands and control lines instead of being attached to the terminal where the process would normally receive commands from the user.

- The user I/O switch user_output is attached to absentee output segment instead of being attached to the terminal as it would in an interactive process environment.

Use the enter_abs_request (ear) command to request absentee processing of commands, jobs, and programs (see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64).

UNDERSTANDING HOW FOREGROUND USAGE DIFFERS FROM BACKGROUND USAGE

Foreground absentee processes enjoy all of the privileges of an interactive user. A foreground absentee process is able to obtain CPU time on demand, and is therefore able to potentially use a great amount of system resources. For information on how to log in a foreground absentee process, see the *Multics Commands and Active Functions* manual, Order No. AG92.

Three kinds of usage are measured for user processes logged in from a terminal or logged in as foreground absentee. These are:

- CPU time
- memory units
- connect time

Usage of these resources is recorded and charged for separately on each shift. The shift schedule is established by the system administrator. Up to eight shifts can be defined; the administrator specifies the shift for each half-hour period of the week.

Background absentee processes are not treated in the same manner as interactive users. Demands for CPU time must wait until system resources are available, depending upon the absentee request's queue.

User processes created as a result of a user's enter_abs_request command can perform background, batch, deferred, or periodic processing. Two kinds of usage are measured:

- CPU time
- memory units

Usage of these resources is recorded and charged for separately in each queue. Usually, four queues are provided. System administrators can also specify which queues run on which shift.

DEFINING ABSENTEE USAGE LIMITS

The number of foreground absentee processes or background absentee processes a project or user can enter is determined by the Max_foreground and Max_background keywords in the System administration Table and Project Definition Table. For foreground absentee processes, the Abs_foreground_cpu keyword controls the amount of cpu time a foreground absentee process can use. The value specified in the SAT for Abs_foreground_cpu, takes precedence over the PMF value. For details on how to set these limits, see Section 32.

MANAGING ABSENTEE LOAD CONTROL

The installation_parms segment contains a number of parameters that enable the site to regulate the background absentee load. (Background absentee jobs are those from the four background absentee queues, numbered 1 to 4, as opposed to jobs from the foreground absentee queue. The latter are regulated by foreground load control.

By default, abs_maxu, the number of absentee slots is a constant for each shift-configuration combination in the configuration table. However, it can optionally be made a function of interactive load. The abs_maxu figure in the configuration table is only used when the parameters described immediately below (pct_abs, min_abs, and max_abs) have not been set.

Background absentee jobs are intended to occupy idle capacity. The number of idle units is computed by the equation:

```
idle_units = max_units - n_daemons - n_interactive
```

where max_units is the system maximum units figure for the current shift and configuration, as specified in the configuration table, and n_daemons and n_interactive are the current number of units occupied by daemon and interactive users, respectively.

Since this figure can fluctuate rapidly as the number of interactive users changes, it functions in the following way: a moving average of idle units over the last M minutes is kept (where M is the value, in minutes, of the idle_time_constant parameter in the installation_parms segment), and the lower of the current value and the average value of idle units is used to compute abs_maxu each time an absentee login decision is made. This algorithm causes the increase in abs_maxu to lag behind a decrease in interactive load, but the decrease in abs_maxu occurs immediately in response to an increase in interactive load.

The abs_maxu figure is computed by the equation:

```
abs_maxu = min (max_abs, max (min_abs, pct_abs * idle_units))
```

that is, abs_maxu varies between min_abs and max_abs, as a function of pct_abs and idle_units. For example, with pct_abs = 10%, max_abs = 6, and min_abs = 1, abs_maxu varies between 1 and 6 as idle_units varies between 10 and 60. If idle_units goes above 60 or below 10, abs_maxu does not rise above 6 or fall below 1.

The three parameters max_abs, min_abs, and pct_abs are kept in the installation_parms segment, and can have different values on each shift.

It is possible to reserve a portion of abs_maxu for the use of the higher priority queues. This avoids delaying jobs from the higher priority queues that are entered into the queues after all the slots have been occupied by jobs from lower priority queues. Two algorithms are described here: the way the qres figures for each queue are computed, and the way all the qres figures are used to determine whether or not an absentee job from an individual queue is permitted to log in.

The number of reserved slots for each queue is computed by the equation:

```
qres(q) = min (max_qres(q), max (min_qres(q), pct_qres(q) * abs_maxu))
```

where qres(q) is the number of slots reserved for queue q. Thus qres(q) varies between min_qres(q) and max_qres(q), as a function of pct_qres(q) and abs_maxu. The min_qres, max_qres, and pct_qres parameters are kept in the installation_parms segment and have separate values for each queue and each shift.

The reserved slot facility does not place an absolute limit on the number of jobs from any (except the lowest priority) queue. It prevents jobs from lower priority queues from logging in if their logging in would leave too few slots available for the higher priority queues, but it allows jobs from high priority queues to occupy all of the slots and exclude jobs from lower priority queues. The algorithm is as follows: to determine if a job from queue q can log in, first reduce the current abs_maxu by the number of jobs currently logged in. This gives the number of available slots. (If it is zero, no job can log in.) Then, for each queue of higher priority than queue q, if the number of jobs currently logged in from that queue is less than qres(i) (where i is the number of that higher priority queue), reduce the number of available slots by the difference. If there is at least one available slot after that has been done, the job can log in.

In addition to the above parameters, there are per-user and per-project limits on the number of concurrent foreground and background processes that a user may have, and limits on the fraction of background absentee slots that users from each load control group may have.

Background Absentee Load Control

Limits can be placed on the number of background absentee slots occupied by users from each load control group. This limit can be a function of the total number of background absentee slots. When using the option (described above) of moving absentee users into different groups for the purpose of placing them in per-queue work classes, the limits described here are imposed before the absentee users are moved to the absentee groups.

The limit on background absentee users from each group is computed by the equation:

```
absentee_limit = min (absentee_max, max (absentee_min,
    absentee_pct * abs_maxu))
```

That is, the limit varies between absentee_max and absentee_min, as a function of absentee_pct and abs_maxu. The parameters absentee_max, absentee_min, and absentee_pct have separate values for each load control group.

If a group has an absolute maximum number of users, then background absentee jobs are counted against this limit. Background absentee jobs can be denied login because of this limit, and they can, by being logged in, cause interactive users to be denied login. Groups that have absolute limits should always be given absentee limits, to prevent background absentee jobs from causing primary interactive users to be denied login.

Foreground Absentee Load Control

Jobs from the foreground absentee queue are governed by foreground load control. Jobs in the foreground queue are treated, for load control and charging purposes, as interactive logins. That is, a foreground job is logged in if the user could have logged in interactively, and while logged in, it occupies a primary slot in the user's load control group. For additional information, see the description of the enter_abs_request command in *Multics Commands and Active Functions* manual, Order No. AG92.

SECTION 37

AUTOMATIC MODE AND UNATTENDED SERVICE

Multics can run in automatic mode and use attended or unattended service. When the system is in automatic mode it should run with unattended service and if the system is in manual mode, it should run with attended service.

DETERMINING WHEN TO RUN WITH UNATTENDED SERVICE

Normally, the system runs in attended service. This means that there is an operator present to mount tapes and disks. If the operator is going to be away from the machine room for an extended period of time, or if the system is going to run during a night or weekend when there is no operator on duty, you should put the system in unattended mode (also called unattended service). When the system is in unattended mode, it appears to the resource control package (RCP) that all mount requests are not honored, and user print requests line up in the queues. Use the "x unattended" command to instruct the system not to run any tapes (since there is no operator available to mount them). The system issues a message that informs users that service is unattended, and sets a flag that instructs the system to reboot itself in the event of a system "crash". If you are running the volume_dumper (or any other dumper), you should mount enough tapes to service it. (The RCP controls and allocates peripheral resources, including tapes, user I/O disks, printers, card readers, and card punches.) If backup is running the operator must preload tapes for it, because he won't be available to mount the tapes when they are needed.

DETERMINING WHEN TO RUN IN AUTOMATIC MODE

Generally, the system runs in manual mode (also called unattended mode). In manual mode, the system pauses in BCE (BOS) after a crash, and waits for the operator to take action. However, if the operator is away from the machine room for an extended period of time or if the system is going to run during a time when no operator will be present, you can arrange to have the system run in automatic mode. Normally, the system should run in attended mode during the hours of heaviest system use to provide the use of all the system facilities to the user community. Off-peak hours such as evenings, weekends, and holidays can be run in unattended mode.

In automatic mode (also called automatic reboot mode), the system automatically takes an fdump and reboots itself after a crash. If the system is in automatic mode, you must have the correct Multics system tape mounted on the correct tape drive and reserved for the reboot. It will not appear to BCE(BOS) that all tape drives are deleted. BCE (BOS) will go to the drive specified in the auto runcom and attempt to boot Multics with whatever tape if finds there. Since manual mode is the usual way of running things, if the operator used the auto runcom to boot Multics, he should always turn off automatic mode as soon as the system is up.

Whenever you put the system in automatic mode you should also put it in unattended service.

For information on how to set and return to unattended and automatic mode refer to the *Operator's Guide To Multics* manual, Order No. GB61.

PART X
MANAGING THE ACCOUNTING SYSTEM

SECTION 38

UNDERSTANDING AUTOMATIC ACCOUNTING OPERATIONS

The disk report calculates disk usage and creates a disk usage report. The accounting administrator can invoke the disk_report command to cause a manual disk usage calculation. Normally, though, disk usage is calculated automatically every night by the absentee job, dodrp.absin, that executes the disk_auto command. The dodrp.absin job is scheduled in queue 1. To run a disk report type:

```
disk_report
```

The system responds with:

```
$ Creating disk usage report.
$ Following figure is total quota/current use
75500/64432
dir: 5500/4432
seg: 70000/60000
charged 906 directories out of 910 to 108 projects

r 1557 1372 1.258 102
```

If you are the system administrator and you intend to invoke the disk_report command as a system administrator rather than an accounting administrator, you should type "ec master disk_report" instead of "disk_report".

The setcrank command schedules the absentee job that performs the accounting segment update. Unless the absentee job crashes or the absentee job queues are lost, there is no need to execute this command. To check whether a job is scheduled, invoke the list_absentee_request command (described in the *Multics Commands and Active Functions* manual, Order No. AG92).

If the accounting update absentee job is not scheduled, type:

```
setcrank
```

The system responds with:

```
3 already requested
r 1557 1.372 1.258 102
```

The crank is a self-rescheduling absentee job that records the daily charges of users and checks for users who should be cut off. Each morning, the accounting administrator should read the output from the crank. Additionally, the accounting administrator should log in and invoke the "day" command to verify that no errors occurred during the running of the crank. The "day" command prints the output from the crank on the terminal and asks if you want to delete the output. Unless there was an error, the accounting administrator should reply "yes" to the question and log out. To use the day command, type:

```
day
```

The system responds with output having the general appearance:

```
Absentee user Accountant.SysAdmin logged in...
r 0330 1.372 1.258 102
Begin charging for  6/30/82 2355.0 to 8/9/82 2345.1
cut 3, warned 7, total charge $45678.90
r 1557 1.372 1.258 102

Absentee user Accountant.SysAdmin logged out...
Delete?
```

The accounting administrator responds with:

```
yes
```

If a system or process error interrupts the crank, it will not run the next night unless the system administrator fixes the problem, completes the interrupted processing in his own process, and resets the abort_crank flag in the value_seg (this flag is the abort_crank variable in >udd>SysAdmin>lib>value_seg).

If the system crashes while running the crank, the abort_crank flag is set to prevent the crank from running again until the state of the accounting segments is determined.

If the system crashes while running the crank, you should examine the absout file produced by the crank. This file indicates what the crank was doing when it was interrupted. Next, check the date and time modified on the segments in >sc1 and the segments in >udd>SysAdmin>admin. Finally, confer with operations to make sure that no files were backed up due to a reload.

To restart the crank you must reset the abort_crank flag. To do this, type:

```
value$set abort_crank false
```

where the "value$set" is the entry point used to change values in the value_seg segment and "abort_crank false" prevents the crank from aborting. This command line allows the crank to run at the next scheduled date/time. To list the current value of the abort_crank flag, type:

```
value$dump abort_crank
```

SECTION 39

MANAGING ACCOUNTING REPORTS/BILLING

The accounting administrator manages the billing and accounting reports. The procedures required for preparing, running, and cleaning up the bill are given below. The accounting reports contained in the >udd>sa>a directory are explained later in this section.

PREPARING THE BILLING

The accounting administrator is responsible for preparing the billing once a month, usually on or before the last day of the month. The bill command takes information from the following segments that are updated during the daily processing. These segments are found in the >udd>SysAdmin>admin directory:

billing_footnote
: This segment is optional. If present, all the text in it is printed at the bottom of each project's usage summary (on the mailing copy only). This segment can be used to announce forthcoming price changes or make other announcements to the administrators for each project. A system administrator must set up the billing_footnote segment. The billing_footnote should be modified just before doing the monthly bill.

disk_stat
: This segment is used in the preparation of the disk usage report. The absentee disk reporting job creates the disk_stat segment.

miscfile
: This segment is the journal for miscellaneous charges and credits associated with all projects. On both the short and long bills, all entries for a particular project are located in this segment and printed. The charge command (described later in the section) creates the miscfile and enters miscellaneous charges into it.

PDTs
: These per-project data bases are segments in >udd>SysAdmin>admin>safe_pdts. These segments are actually copies of the segments in >sc1>pdt. Each one contains the complete usage data for each user for the billing period. These segments are prepared by the daily accounting job (called the crank).

projfile
: This segment is used to obtain the disk usage figures stored in it by the disk_report command. The project title and the name and address of the project supervisor are also used to create headings on the long and short bills.

reqfile
: This segment contains the charges that are actually billed. Daily processing has updated the reqfile segment from the figures in the PDTs, so the two should agree. (If they do not agree, an error comment is printed and the reqfile values are used.) The name and address of the person in charge of the account, as well as requisition numbers, amounts, and cutoff dates, are also extracted from this segment.

today usage totals (>udd>sa>a>today.use_totals)
A statistical data base that describes month-to-date system resource availability and usage: prepared daily from meters_data and the PPTs. This segment is used in the preparation of the usage summary report (system_month.report).

yesterday usage totals (>udd>sa>a>yesterday.use_totals)
The previous day's copy of today.use_totals. This segment is used in the preparation of the usage summary report (system_month.report).

For more information on these segments refer to Overview of System Tables/ Data Bases in this manual.

BILLING

The procedure for billing consists of the following steps:

- Preparing the bill
- Running the billing programs
- Accepting the bill
- Cleaning up the bill once it is accepted

The registration and accounting administrator who is in charge of the billing should use the entry points to the billing commands described below. The accounting administrator is constrained in the special command system and can perform only a certain specific set of accounting functions. Additionally, the accounting administrator does not have access to the ec.master table.

If you are a system administrator and you are in charge of the billing, you can enter a billing command by changing your working directory to >udd>SysAdmin>admin and using the ec master command in combination with the selected billing command as in the example below:

```
ec master bill delete
```

Preparing the Bill

The amount of storage required by the bills varies, depending on the number of users registered on the system and the number of projects. It may come to several thousand disk records.

To prepare a bill, the accounting administrator uses the following command:

```
bill prepare
```

This command calls the biller.ec segment (in the directory >udd>SysAdmin>lib) to perform billing operations. The preparation phase also consists of checking to see that a disk usage report has been run recently. The accounting administrator should also ensure that the billing_footnote segment is up-to-date and that all miscellaneous charges have been input. To print the contents of the miscfile, use the pmisc command (described later in this section). This command is useful as a check to ensure that all entries in the miscfile are correct before the bills are run. The pmisc command is described later in this section.

Use the following bill_prepare command to prepare a bill:

```
ec.master bill_prepare
```

This command performs the same function described above.

Running the Billing Programs

The accounting administrator should ensure that there is enough quota before starting the billing run. Use the get_quota command to obtain this information. The accounting administrator initiates the actual running of the bill by typing:

```
bill run MM DD YY
```

where "MM DD YY" is the date on which the billing is being run--ideally the last day of the month.

If you are the system administrator and you want to run the bill you would type:

```
ec.master bill run MM DD YY
```

The command performs the same function as described above.

Provision is made for the output of billing information to be tailored to the needs of the individual site. To use this feature, write one or more programs, or modify the existing programs, to produce the information in the format needed at the site. He must also modify the bill command (in biller.ec) to execute this program. Type:

```
bill run MM DD YY arg
```

where "MM DD YY" are as above and "arg" is an argument accepted by the site_dependent program that produces billing output in the desired format. The accounting administrator should not supply the "arg" argument unless directed to do so by the system administrator.

The run processing produces all of the reports and bills described below. It dprints one copy of the bills in the highest queue before starting on the usage summary report.

bill

This report is a listing, by account number, of the charges made to each account. One or more projects can be billed with the same account number. One line is printed for each project, showing the charges this month and the charges to date, face amount, and requisition balance.

diskreport
This report shows each project's disk usage for the month. It also has a map of every directory in the hierarchy that has a disk quota. This map gives the project's current usage, charge for the month, and quota.

long_bill
This report is intended for retention and filing. It contains the same complete information as the mailing_copy report without the footnotes, distribution page, and list of current prices.

It is a complete breakdown and justification of charges for each project. For each project, the bill has from one to five sections:

1. Charge summary, by user

2. Interactive usage, by user

3. Absentee usage, by user

4. I/O daemon usage, by user

5. Other charges (tape, high-speed line, etc.), by user

In addition, the detail summary shows the project's disk and miscellaneous charges, lists the current prices, and may have (as a footnote) a message to all project supervisors. A distribution page is produced before each project's bill.

mailing_copy
This report is a copy of the long bill with the footnote and distribution page attached to it.

miscs.print
This report is a listing of all miscellaneous charges and credits for the month.

msum
This report is the monthly summary which is totaled by account number. The account number can have different requisitions associated with it. If the account number is different, it is placed on a separate line. The msum report has one line per requisition or purchase order and gives the same information as in the bill report.

short_bill
This report consists of just the user charge summaries by project from the long_bill. A grand total page is printed which is a summary of system usage.

system_month.report
This report gives a summary of Multics usage for the billing period. The first page of this two-page report displays information such as the number of logins this month versus last month. The second page displays information such as the maximum load units and the type of software running on a particular day.

If the accounting administrator finds errors (such as negative values) in the bill, he should contact the system administrator immediately.

Accepting the Bill

If the accounting administrator does not find any errors in the bill, he should invoke the "bill accept" command by typing:

```
bill accept arg
```

where arg can be the name of a month, a Julian date, or any name that uniquely identifies the billing run.

The "bill accept" command does the following:

1. Dprints copies of the billing output and reports for administrators and project supervisors.

2. Copies the segments used to create the bill into the >udd>SysAdmin>admin>HF directory. The names of these segments are prefixed by the argument specified to the "bill accept" command (e.g., the reqfile segment is copied into the arg.reqfile segment).

3. Resets the accounting data bases for the next month.

4. Resets the disk meters in the directory branches.

5. Reinitializes the monthly statistics printed out in the daily and monthly reports.

The system administrator invokes the bill accept command by typing:

```
ec.master bill accept
```

This command performs the same functions as above.

Cleaning Up the Bill

After the bills have been mailed out, the accounting administrator should use the bill delete command to delete the bills from the disk. The bill delete command deletes the current months billing segments from storage.

To use the bill delete command, the accounting administrator types:

```
bill delete
```

To use the bill delete command, type:

```
ec.master bill delete
```

This command performs the same function as above.

MANAGING THE MISCFILE

The system administrator can manage the miscellaneous file by entering charges and credits into the miscfile and deleting charges and credits from the miscfile. The system administrator can also print the contents of the miscfile. All of these commands are explained below and in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Printing the Contents of the Miscfile

To print the contents of the miscfile, use the pmisc command. This command is useful as a check to ensure that all entries in the miscfile are correct before the bills are run. The accounting administrator can print the miscellaneous charges and credits for an individual project or he can print the entire miscfile. He can also print the charges and credits for specific dates. The dates must be in the form mm/dd/yy or mm/dd. If more than one date appears on the same line, they must be separated by spaces. Type "x" instead of a Project_id to exit from the command. To print the entries for the Alpha project incurred on July 9, type:

```
! pmisc
  project
! Alpha
  date
  7/9/84 7/12/84
  7/09/84 23 Alpha 4.5.0 manual
  7/12/84 32 Alpha 7.60 manual
  project
! x
  r 1557 1.234 1.001 115
```

To print all the entries in the miscfile, type:

```
! pmisc
  project
! all
  dates
! all
  07/01/84 1 Gamma 23.56 manuals
    .
    .
    .
  07/29/84 207 Beta .50 news bulletin
  r 1557 1.234 1.001 115
```

Entering and Deleting Charges

To enter miscellaneous charges into the miscfile, use the charge command. For each transaction, the Project_id, the amount, and an explanation are required. All three input items for a transaction can be put on the same line or they can be supplied, one at a time. To exit from this command, type "x" instead of a Project_id.

To charge the Alpha project for some manuals, type:

```
charge
project
Alpha
amt
10.55
explanation
manuals ordered 6/23 Jones
project
x
r15571.372 1.258 102
```

To delete charges from the miscfile, use the dmisc command. For each transaction, the project_id and number of the mscfile entry are required. The number of the mscfile entry is printed using the pmisc command. Both input items for a transaction can be put on the same line or they can be supplied one at a time. To exit from this command, type "x" instead of a Project_id.

To delete charges from the alpha project, type:

```
dmisc
project
Alpha
number
23
project
x
r 1557 1.372 1.258 102
```

Entering and Deleting Credits

To enter miscellaneous credits into the miscfile use the credit command. For each transaction, the Project_id the amount, and an explanation are required. All three input items for a transaction can be put on the same line or they can be supplied one at a time. To exit from this command, type "x" instead of a Project_id.

To credit the Alpha project for a crash, type:

```
credit
project
Alpha
amt
23.00
explanation
system crash 6/23 Smith
project
x
r 1557 1.372 1.258 102
```

To delete charges from the miscfile, use the dmisc command. For each transaction, the Project_id and the number of the miscfile entry are required. The number of the mscfile entry is printed using the pmisc command. Both input items for a transaction can be put on the same line or they can be supplied one at a time. To exit from this command, type "x" instead of a Project_id.

```
! dmisc
  project
! Alpha
  number
! 23
  project
! x
  r 1557 1.372 1.258 102
```

Creating a Project Month-To Date Report for a Project

To obtain a month-to-date report for a project, use the command proj_mtd with the Project_id. To use this command, type:

```
proj_mtd Project_id
```

where "Project_id" is the name of the project whose mtd report is to be printed.

The proj_mtd command types a month-to-date report for a particular project's usage. The report lists all users on the project and their month-to-date dollar totals, as well as disk and miscellaneous charges.

You can run this command anytime during the month. For more information on this command refer to the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

UNDERSTANDING ACCOUNTING REPORTS

The following accounting reports are found in the >udd>sa>a directory.

Daily Account Status Summary Report (sumry)
: a one line summary by project showing the account number and the requisition number, prepared daily.

Daily Cutoff Account Report (cutrpt)
: a report of accounts that have been cut off daily or accounts that have been warned that they are reaching their dollar limit. This report is prepared daily.

Daily Disk Statistics/Usage Report (diskreport)
: a report of projects disk usage, prepared and dprinted daily by the dodpr.absin job.

Daily Usage and Revenue Report (usage_and_revenue.report)
: a segment containing the output of the usage_and_revenue command; produced daily by the crank; suitable for printing.

Daily System Statistical Report (system.report)
the daily statistical report; first part of daily_log_N.

Weekly Black and White Chart
the system "black and white" chart, showing system availability for the week; printable report.

Weekly Report (weekly.report)
a weekly summary of administrative operations

Monthly Long Bill
this report is intended for retention and filing. It contains the same complete information as the mailing_copy report without the footnotes, distribution page, and list of current prices.

Monthly Short Bill
this report consists of just the charge summaries from the long_bill. The prices and footnotes are suppressed. A grand total page is printed. This report is intended for the use of system administrators and others who find the long_bill too bulky.

Monthly Account Bill
this report consists of a one page report for each separate account listing the projects that are charged to that account.

Monthly Charge Summary
this is the monthly summary report consisting of a one line report for every account number that is being billed.

Monthly Miscellaneous Charges Summary
this is the final monthly printing of the miscellaneous file before it is cleared for the next month.

Monthly Disk Report
the last daily disk report run at the end of the month. The absentee job dodrp.absin actually does the running and produces the disk report.

Monthly Usage and Revenue Report (monthly_usage_and_revenue.report)
a segment containing the monthly usage and revenue report; produced by monthly billing.

Monthly Black and White Chart (bwchart.print)
the system "black and white" chart, showing system availability for the month; printable report.

Billing Footnote (billing_footnote)
a segment of announcements (e.g., forthcoming price changes) to project supervisors that is printed at the bottom of each project's usage summary report (on the mailing copy only); this segment is optional.

PART XI
SETTING OTHER SYSTEM OPTIONS

SECTION 40

MANAGING SETTABLE PARAMETERS

This section is concerned with the parameters contained in the installation_parms segment. These parameters require attention when the site is set up and then require relatively little change thereafter.

UNDERSTANDING INSTALLATION_PARMS

Most site parameters used in the operation of the administrative system are kept in >sc1>installation_parms. When the accounting start up exec_com (acct_start_up.ec) is executed during start up at a new Multics site, it automatically sets these parameters to default values. The system administrator uses the ed_installation_parms command both to read current values and to alter these values in the installation_parms segment. A discussion of each parameter follows.

It is recommended that if you are reading this information for the first time, invoke the ed_installation_parms command to print out the >sc1>installation_parms segment to be used for reference.

The installation_parms segment contains the site-dependent 32-character site identification (ID) field, typed out when a user dials up, and the site titles, used at the top of each page after monthly billing and in many other reports. The site ID contains the company and department abbreviation along with the city and state where the Multics site is located. An example of a site ID is:

```
installation_id:  Multics Service, Company,City,State
```

The site titles consist of two strings, the department ID and the company name. Each string is entered twice, once single-spaced and once double spaced. Examples of these titles as they appear in the installation_parms segment are:

```
company:          Company Name, Inc.
department:       Computer Center
companyids:       C o m p a n y   N a m e,   I n c.
departments:      C o m p u t e r   C e n t e r
```

MODIFYING INSTALLATION_PARMS

Use the ed_installation_parms command both to read current values and to alter these values in the installation_parms segment. The values that can be edited by the ed_installation_parms command are listed below. For detailed information about the ed_installation_parms command, see the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Default Absentee CPU Time Limit/Queue

The default absentee CPU time limit is specified with the abs_cpu_default_limit keyword using the ed_installation_parms command. These default limits are used when no time limit is specified by the user who enters a job in one of the background absentee queues. The limit is given in seconds. The default value for each queue is 1200 seconds (20 minutes).

Maximum Absentee CPU Time Limit/Queue

The installation_parms segment also contains the maximum absentee CPU time limit for absentee jobs, from each of the four background absentee queues, on each shift. These are specified by the abs_cpu_max_limit keyword of ed_installation_parms.Jobs whose CPU time limits (specified by the submitter) exceed the limits for the current shift are automatically deferred to a shift with high enough limits. This allows the site to prevent long-running jobs from logging in during peak load periods. The default value for all queues on all shifts is 1200 seconds (20 minutes).

Default Absentee Queue

The default absentee queue is used for submission of jobs by the enter_abs_request, pl1_abs, etc. This parameter may have a value in the range 1 to 4 and defaults to 3 unless you change it using the abs_default_queue keyword of ed_installation_parms.

Absentee Scheduling Priority

The scheduling priority for a Multics process is calculated by the scheduler using a complex algorithm that takes several factors into account. Each process is assigned a value for timax when it is created. This value limits the depth to which the process may sink in the scheduling queues. The timax value for background absentee processes is obtained from the the installation_parms segment, and may be different for different queues. A larger number results in a lower scheduling queue. By default, all background absentee queues are set to have a timax of 16000000 microseconds while interactive and foreground absentee processes get a timax of 8000000. This means that background absentee jobs sink rapidly to the bottom of the scheduling queues, where they are given relatively long CPU quanta, and that interactive and foreground absentee users always get better response than background absentee users. By changing the default values associated with the abs_timax keyword for a certain queue, a site may experiment with other values for the absentee timax.

Access Ceiling

The highest authorization allowed on the system (system_high) is specified by the access_ceiling keyword. This keyword defines the highest numerical sensitivity level allowed and specifies all access categories used. For the system to operate correctly, all sensitivity levels numerically less than or equal to the level given with the access_ceiling keyword, and all categories contained in the category set of the access_ceiling keyword, must have both a long and a short name. Installation of a PNT, SAT, CDT, or PDT is refused if any entry specifies a level or category greater than one given with the access_ceiling keyword.

It is often convenient to give a null string (i.e., blank) for both the long and short names of the lowest sensitivity level (level 0). If this is done, processes with the default (system_low) authorization see no visible evidence of the nondiscretionary access controls. The default values for the access_ceiling parameter are:

```
access_ceiling:      level=0, categories=000000
```

The default values for the level_names, category_names, and access_ceiling keywords are assigned when the accounting start up exec_com is run; they define names for all levels and categories, but prevent use of authorizations greater than system_low. Either a system administrator or a system security administrator may alter these default values.

Authorization Names

Nondiscretionary access control keywords (AIM features) are also kept in the installation_parms segment. (See "Access Control" in the *Multics Programmer's Reference Manual*, Order No. AG91 for further details on nondiscretionary access controls.) From 1 to 8 sensitivity levels and from 0 to 18 access categories may be defined. Each sensitivity level and each access category is defined by assigning it both a long and a short name. The long name may be up to 32 alphanumeric characters in length and may include spaces. The short name may be up to eight alphanumeric characters long and may not contain spaces. (It is recommended that the short names be easy to remember as a convenience for users when they log in.) Each name must be unique among all level and category names. However, the long and short names of a given level or category may be identical. Level names and category names may be read and modified by specifying the level_names and category_names keywords with the ed_installation_parms command. Default values for these keywords are:

```
level_names:     level_N, lN (where N is from 1 to 7) and
                    "" (for level 0)

category_names: category_N, cN (where N is from 1 to 18)
```

Configuration Table

A configuration table is kept in the installation_parms segment that describes each of the system configurations that a site uses, by the number of CPUs, number of pages of memory, number of pages of bulk store, and shift. For each such entry, the table contains the value of the maximum number of load units allowed on the system, the maximum number of absentee users allowed, the highest-numbered (lowest-priority) absentee queue from which jobs are run, and two response control keywords (high and low). Unless automatic adjustment of maxunits has been disabled (see the description of the maxu command in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64), this table is referenced to set maxunits whenever the system is brought up, whenever the shift changes, and whenever the system configuration changes.

If response control is enabled, the system attempts to adjust maxunits on every login and logout, so that the average queue length falls between the low and high values. Maxunits is increased whenever the average scheduler queue length is below low; it is decreased whenever the queue length is above high; and it is set to the current number of users whenever the queue length is between low and high.

The order of the elements in the configuration table is significant. The elements should be in order by number of CPUs, within that by amount of main memory, within that by amount of bulk store. and within that by shift; smaller numbers first. The lookup continues as long as any table element is less than the value being looked up, so that an element specifying very large numbers stops the lookup. This feature can be used if, for instance, the site does not wish to have different keywords depending on shift: instead of having eight table entries for each combination of CPU and memory for shifts 0–7, the system administrator may supply only one entry, which has a shift number of 7. The config_table keyword of the ed_installation_parms command automatically maintains the configuration table in its properly sorted order.

Count Parameter of Wakeup Loop Detector

The count parameter of the terminal channel wakeup loop detector is specified with the cwe_count keyword of ed_installation parms. If a user causes more than cwe_count interactions within cwe_time while logging in, the channel is hung up. An interaction occurs every time the Answering Service is notified of an event for the terminal channel. The number of interactions perceived by the Answering Service is typically the same as the number of input lines (terminated by a newline character) entered by the user. The default value for this keyword is 10.

Time Parameter of Channel Wakeup Error Loop Detector

The time parameter of the channel wakeup error loop detector in seconds is specified in ed_installation_parms using the cwe_time keyword. Default time is 3 seconds. See also the description of cwe_count above.

Directory and Segment Quota

The default_pdir_seg_quota keyword and the default pdir_dir_quota keyword of ed_installation_parms let you set the default segment quota and the default directory quota for a process, respectively.

Device Names

Device names are the names of the miscellaneous devices. The device_names keyword of ed_installation_parms lets you change the name of an existing device or print the names of all the defined devices. Changing a device name to devN, where N is a number from 1 to 16 indicating the location of the device in the table. has the effect of deleting the device, since unused device table locations contain names of that form. If a deleted device is the last one in the list, the list is shortened by one.

Charging for certain devices is built into the system. These devices must be defined in the device table, with their names spelled exactly as the system expects to find them. The required device names are:

- tape
- tape_mt
- disk
- disk_mt
- lv

They control prices for tape volumes, assigned to a process, tape drives, disk volumes, disk drives, and private logical volumes attached to a process. These devices are defined automatically by the default request. A warning message is printed by the print device_prices and print keywords for any of these devices that is not defined.

Device Prices

The device_prices keyword of the the ed_installation_parms command let you list the names and prices (dollars per hour, for each shift) for each of up to 16 miscellaneous devices (teletype channels, high-speed channels, tapes, etc.). For the print request, if the name of a device is typed ahead on the input line after the device_prices keyword, the prices for that device are printed; otherwise the prices of all devices are printed. For the retype request, the name of each device is typed and new prices are requested for all shifts. The default is $1.00 per hour for each device type on all shifts.

Default CPU Time Limit for Foreground Absentee Queue

Jobs from the foreground absentee queue are governed by foreground load control. Jobs in the foreground queue are treated, for load control and charging purposes, as interactive logins. That is, a foreground job is logged in if the user could have logged in interactively, and while logged in, it occupies a primary slot in the user's load control group. For additional information, see the description of the enter_abs_request command in the *Multics Commands and Active Functions* manual, Order No. AG92.

The foreground_cpu_default_limit keyword of ed_installation_parms specifies the default cpu limit associated with foreground jobs. The default value is 1200 seconds (20 minutes).

Count Parameter for Fatal Process Loop Detector

The fpe_count keyword of ed_installation_parms specifies the count parameter of the fatal process loop detector. If a process has more fatal process errors than the value specified for fpe_count within the limit specified by fpe_time, it is considered to be a fatal process error loop and the process is logged out. The default count is 3.

Time Parameter of Fatal Process Loop Detector

The fpe_time keyword of ed_installation_parms is used to manipulate the time keyword of the fatal process error loop detector. Default is 60 seconds (see the description of fpe_count, above).

Idle Time

Idle time is the time over which moving average of foreground load is taken, for use in adjusting maximum background absentee users (abs_maxu). The idle_time_constant keyword can be used to set this parameter in seconds. Default is 900 seconds (15 minutes).

Inactive Time

Inactive time is the number of real-time seconds a process may remain blocked before being bumped for inactivity. It is controlled by the inactive_time keyword of the ed_installation_parms command. The default is 3600 seconds (60 minutes).

Installation Identification

The installation identification is the installation name, city, and state (maximum of 32 characters), for example: Company, City, State. The installation_id keyword sets this parameter. The default is "Installation and location".

Log Parameters

Log parameters specify the number of pages which may be written into the ring 0 syserr log before being copied. It can be any value from 1 to one-half the size of the log partition (typically 1 to 121 pages). A value of 0 means use the default value. Log parameters can be manipulated with the log_parameters keyword of ed_installation_parms.

You may choose from the following values.

0
: Enable log copying with a default threshold. The ring 0 syserr log is copied into ring 4 at every shutdown, and whenever there are 10 or more uncopied pages. This is appropriate for most sites.

N(1<=N<=127)
: Enable log copying. The ring 0 log is copied into ring 4 at every shutdown, and whenever N pages of new messages have accumulated. A small value (around 4) is appropriate for sites using protection auditing and for development sites where the print_sys_log and heals_report commands are used frequently.

Log-In and Log-Out Parameters

The maximum time allowed for a login attempt, the cycle time between initializer accounting updates, the maximum inactive time before automatic logout, the amount of time between the warning and the logout for a user, and the rate of interactions during login are all specified in seconds in the installation_parms segment. The number of login tries a user may have before the system hangs up on him is also specified there.

The log-in keywords and their usual values are:

Default	Range	
login_time:	360	180-360 seconds (more if noisy lines)
update_time:	900	900-5400 seconds
inactive_time:	3600	900-5400 seconds
warning_time:	300	180-300 seconds
cwe_count:	10	1-unlimited seconds
cwe_time:	3	1-unlimited seconds
tries:	6	1-6

These values may be modified according to the needs of the site; however, the suggested ranges are reasonable limits. For example, the accounting update interval should not be too small, or else initializer cpu usage goes way up; it should not be too large (unless the system hardly ever crashes), or else revenue lost by a system crash increases. (Note that the output format of some accounting programs is based on a 15 minute (900 second) accounting update interval.)

New Process Authorization Level

The new_proc_change_auth keyword to ed_installation_parms controls the use of the -auth control argument ot the new_proc command. The -auth control argument to the new_proc command lets users create a new process at a different access authorization. This feature should be disabled at a security conscious site.

Operator Inactivity Limit

The operator_inactivity_limit keyword of ed_installation_parms lets you control the time limit (in seconds) imposed on operator inactivity. Operators who enter no input for the specified number of seconds, are automatically signed out.

Operator Login

The require_operator_login keyword of ed_installation_parms controls the operator identification and authentication option.

Password Control

There are several keywords in ed_installation_parms that let you control user passwords. They are password_change_interval, password_expiration_interval, password_gpw_length, and password_min_length. For details on each of these keywords, see the *Administration, Maintenance and Operations Commands manual* Order No. GB64.

Percent of Idle Units Available to Background Absentee Jobs

The installation_parms segment contains a number of keywords that enable the site to regulate the background absentee load. (Background absentee jobs are those from the four background absentee queues, numbered 1 to 4, as opposed to jobs from the foreground absentee queue. The latter are regulated by foreground load control.

By default, abs_maxu, the number of absentee slots (i.e., the maximum number of concurrently logged in background absentee jobs) is a constant defined for each shift-configuration combination by the abs field in the configuration table (see below). However, it can optionally be made a function of interactive load. The abs_maxu figure in the configuration table is only used when the keywords described immediately below (pct_abs, min_abs, and max_abs) have not been set.

Background absentee jobs are intended to occupy idle capacity. The number of idle units is computed by the equation:

```
idle_units = max_units - n_daemons - n_interactive
```

where max_units is the system maximum units figure for the current shift and configuration, as specified by the max field in the configuration table, and n_daemons and n_interactive are the current number of units occupied by daemon and interactive users, respectively.

Since this figure can fluctuate rapidly as the number of interactive users changes, it is smoothed in the following way: a moving average of idle units over the last M seconds is kept (where M is the value, in seconds, of the idle_time_constant keyword in the installation_parms segment), and the lower of the current value and the average value of idle units is used to compute abs_maxu each time an absentee login decision is made. This algorithm causes the increase in abs_maxu to lag behind a decrease in interactive load, but the decrease in abs_maxu occurs immediately in response to an increase in interactive load. The default idle_time_constant is 900 seconds (15 minutes).

The abs_maxu figure is computed by the equation:

```
abs_maxu = min (max_abs, max (min_abs, pct_abs * idle_units))
```

that is, abs_maxu varies between min_abs and max_abs, as a function of pct_abs and idle_units. For example, with pct_abs = 10%, max_abs = 6, and min_abs = 1, abs_maxu varies between 1 and 6 as idle_units varies between 10 and 60. If idle_units goes above 60 or below 10, abs_maxu does not rise above 6 or fall below 1.

The three keywords max_abs, min_abs, and pct_abs are kept in the installation_parms segment, and can have different values on each shift. By default, they have values that disable dynamic adjustment of abs_maxu.

Percent of Absentee Slots Reserved for Each Queue

It is possible to reserve a portion of abs_maxu for the use of the higher priority queues. This avoids delaying jobs from the higher priority queues that are entered into the queues after all the slots have been occupied by jobs from lower priority queues. Two algorithms are described here: the way the qres figures for each queue are computed, and the way all the qres figures are used to determine whether or not an absentee job from an individual queue is permitted to log in. By default, the reserved queue mechanism is disabled.

The number of reserved slots for each queue is computed by the equation:

```
qres(q) = min (max_qres(q), max (min_qres(q), pct_qres(q) * abs_maxu))
```

where qres(q) is the number of slots reserved for queue q. Thus qres(q) varies between min_qres(q) and max_qres(q), as a function of pct_qres(q) and abs_maxu. The min_qres, max_qres, and pct_qres keywords are kept in the installation_parms segment and have separate values for each queue and each shift.

The reserved slot facility does not place an absolute limit on the number of jobs from any (except the lowest priority) queue. It prevents jobs from lower priority queues from logging in if their logging in would leave too few slots available for the higher priority queues, but it allows jobs from high priority queues to occupy all of the slots and exclude jobs from lower priority queues. The algorithm is as follows: to determine if a job from queue q can log in, first reduce the current abs_maxu by the number of jobs currently logged in. This gives the number of available slots. (If it is zero, no job can log in.) Then, for each queue of higher priority than queue q, if the number of jobs currently logged in from that queue is less than qres(i) (where i is the number of that higher priority queue), reduce the number of available slots by the difference. If there is at least one available slot after that has been done, the job can log in.

In addition to the above keywords, there are per-user and per-project limits on the number of concurrent foreground and background processes that a user may have, and limits on the fraction of background absentee slots that users from each load control group may have.

Prices

Prices can be set for the following using the prices keyword in ed_installation parms:

- disk storage (in units of dollars per page-seconds or page-month)
- per-month user registration
- per-shift CPU time (in units of dollars per virtual CPU hour)
- per-shift connect time (in dollars per hour of terminal connect time)
- per-shift terminal I/O operations (in dollars per 1000 lines of terminal I/O)
- per-shift memory usage (in dollars per 1000 memory units)

Any site may, of course, set the prices for some resources to zero and use other administrative means to control resource usage. By default, disk storage is charged at $.50 per page-month; there is no per-month user registration fee; CPU time is $240 per virtual CPU hour; connect time is $1.25 per hour; there is no terminal I/O charge; memory usage is $15 per 100 memory units.

Changes to prices (and additions to device and resource names) may not be effective until the next answering service startup (i.e., system bootload). Various software modules take "snapshots" of installation_parms during initialization to increase their operating efficiency and are therefore locked in to that set of values until shutdown.

It is recommended that rate changes only be made at the end of a month, just after bills have been run. In a special session, make an extra run of the diskreport and the crank, then run bills and set the new rates before allowing any users to log in. This avoids discrepancies between a user's charges and the user's resources figures multiplied by the current rates. Such discrepancies are upsetting both to users and to programs in the accounting system.

Queue Prices

Queue prices for the following can be controlled with the queue_prices keyword in ed_installation_parms:

- absentee virtual CPU time
- absentee memory usage
- I/O daemon usage

Use the queue_prices keyword to change the prices for absentee virtual CPU time (dollars/virtual CPU hour), absentee memory usage (dollars/1000 memory units), and I/O daemon usage (dollars/1000 lines), for up to four queues. By default, the prices are $200 per virtual CPU hour, $15 per 100 memory units, and $1.80 per 1000 lines printed.

Resource Prices

The resource_prices keyword of ed_installation_parms provides the names and prices of resources for which users are to be charged. The resource price list is currently used only for charging for special forms in I/O daemon requests. For the print request, if the name of a resource price is typed directly after the resource_price keyword, that price is printed; otherwise all the prices are printed. For the retype request, the name of each resource is typed and a new price is requested. By default, no resource names or prices are defined.

Resource Names

Resource names, described with the resource_names keyword in ed_installation_parms are names of existing entries in the resource price list. This keyword permits changing the name of an existing resource.

Resource Wait Time

The resource_wait_time keyword of ed_installation_parms specifies the time interval (in seconds) at which to keep checking for resource availability when an absentee job is waiting for a resource and there are no other jobs logging in or out. Default is 300 seconds (5 minutes).

RCPRM Flags

Several flags are maintained in the installation_parms segment and are used by RCP at initialization time to determine certain actions to be taken during RCP operation. These flags and their default values are:

```
authentication_level      nominal
auto_registration         off
rsc_mgmt_enabled          off
unload_on_detach          off
```

For more detailed information on each of these keywords, see the description of the ed_installation_parms in the *Multics Administration, Maintenance and Operations Commands* manual, Order No. GB64.

Automatic Volume Detachment

The unload_on_detach keyword of ed_installation_prams command causes detached volumes to be unloaded automatically. The string "on" sets the flag on; "off" is the default. When the flag is off a detached volume remains on its drive until the drive is needed for another volume. This flag has no effect when the user specifies "-retain all" in the attach description of the volume being detached (see the tape_ibm_ module). This keyword may only be used with the change, print, or retype requests.

Default Security Level for Volume Authentication

The default security level for volume authentication can be manipulated with the authentication_level keyword in installation_parms. It is default level of security for volume authentication under RCP Resource Management. Acceptable values for this keyword are:

- none
- nominal
- automatic

For details, refer to the ed_installation_parms command in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Automatic Volume Registration

Automatic volume registration is permitted under RCPRM. Automatic registration is controlled with the auto_registration keyword in installation_parms and can be manipulated with the ed_installation_parms command. Automatic registration of an unregistered volume is performed when the operator allows a user to mount the volume. The volume is registered, with default attributes, to the user requesting the volume. This keyword may only be used with the change, print, or retype requests.

Shift Table

System usage may be divided into a maximum of eight shifts, numbered from 0 to 7. These shifts may begin at any half-hour during a week; the shift is the same at the same hour during every week for every user. The operator command, shift, is used for overriding the regular shift mechanism on holidays. Shifts are set by filling in a table that contains 336 entries (one for each half-hour in the week) with a digit from 0 to 7. You must specify the shift_table keyword with the ed_installation_parms command to inspect or modify this table. The default shifts 1, 2, 3, and 4 are:

```
shift 1   0800-1800 weekdays
shift 2   1800-2400 weekdays
shift 3   0000-0800 every day
shift 4   0800-2400 weekends
```

If the segment >sc1>shift_config_change.ec exists, it is executed by the answering service at each shift or configuration change. It can contain commands to alter system keywords or take any other actions required by the site.

Real-Time Limit for Suspended Process

The sus_real_time keyword of the ed_installation_parms command lets you set the real time limit imposed on a suspended process. The default is 180 seconds. Before going blocked, a suspended process sends a wakeup to the answering service, specifying the event channel on which it is about to go blocked, and over which it can be released. If the process fails to respond in this way within the specified interval of real time, it is destroyed.

CPU Time Limit for Suspended Process

CPU time limit imposed on a suspended process can be manipulated with the sus_cpu_time keyword of the ed_installation_parms command. The default is 5 seconds. In a user process, the default handler for the sus_ signal goes blocked and waits to be released. If the process fails go blocked, and continues running, it is destroyed after it has used the specified amount of CPU time (after the sus_ signal is sent).

Titles

The site titles consist of two strings, the department ID and the company name. Each string is entered twice, once single-spaced and once double spaced. Titles can be changed with the titles keyword of the ed_installation_parms command.

Examples of these titles as they appear in the installation_parms segment are:

```
company:        Company Name, Inc.
department:     Computer Center
companyids:      C o m p a n y   N a m e,   I n c.
departments:    C o m p u t e r   C e n t e r
```

Real-Time Limit for Terminating a Process

The real-time limit for terminating a process is the limit imposed on a process that has been sent a trm_ signal. Default is 120 seconds. This value can be changed using the trm_real_time keyword of the ed_installation_parms command.

A process being terminated involuntarily by the answering service (e.g., by the bump command), as opposed to a process that terminates itself voluntarily (e.g., by the logout command), is sent a trm_ signal, and is given a small amount of time to terminate itself. The default handler for trm_ signals the finish condition in the user's process.

CPU Time Limit for Terminating a Process

The CPU time limit for terminating a process is the cpu time limit imposed on a process that is being terminated and has been sent a trm_ signal. Default is 5 seconds. This value can be changed using the trm_cpu_time keyword of the ed_installation_parms command.

Login Tries

The tries keyword of ed_installation parms specifies the number of login tries a user may have before the system hangs up. This value can be from 1 to 6.

Accounting Update Interval

The accounting update interval, manipulated with the update_time keyword of ed_installation_parms, controls the number of real-time seconds between system accounting updates.

Validating Daemon Commands

The validate_daemon_commands keyword of ed_installation_parms controls activation of the secure daemon facility. For details, see the *Administration, Maintenance and Operations Commands manual* manual, Order No. GB64.

Warning Time

The warning_time keyword of ed_installation_parms lets you control the number of real-time seconds between the warning of an automatic logout of a process and the actual logging out of the process.

In the following example, the system administrator first prints the current value of the "warning time" variable (the number of seconds before shutdown that a user is warned), and then changes the value. Finally, the change is made to (Lines, or portions of lines, typed by the administrator are preceded by "!").

```
!  ed_installation_parms
   type  !p
   id    !warning_time
   warning_time:  300 sec.
   type  !c
   id    !warning_time
   warning time (secs)  !  6000
   type  !w
   type  !q
   <ready>

!  ed_installation_parms
   type  ! c shift Fri 18 00 thru Mon 07 30 5
   type  ! w
   type  ! q
   <ready>
```

This defines shift 5 to start at 1800 Friday and end at on 0800 Monday.

REBOOTING TO HAVE CHANGES TAKE EFFECT

If you use ed_installation_parms to make changes to default installation_parms values, for those changes to be valid you must reboot the system.

UNDERSTANDING VALUE_SEG

Values to be used in several administrative exec_com segments to supply site-dependent values as command argument are kept in the value_seg segment located in >udd>SysAdmin>lib. The values you enter in the value_seg can eliminate the need for many site modifications to the exec_com segments.

MODIFYING VALUE_SEG

To enter or modify values in the value_seg, use the value_set command. The following values are defined by default:

disk_time
: the time of day that the disk accounting runs (this should be before the crank).

abort_crank
: "true" if crank has aborted, else "false".

last_crank
: the date and time of the last crank.

last_diskreport
: the date and time of the last disk report.

weekly_time
: the time for the next run of the weekly report. This is a time string, in the form "2300. 6 days."

XXX_dest
: a string passed as the argument to the -destination control argument of the dprint command. Some of the valid strings are:

  directorN
  : installation director (where 0<N<7)

  adminN
  : system administrator (where 0<N<1)

  assuranceN
  : system assurance (where 0<N<1)

  sysprgN
  : system programming (where 0<N<2)

XXX_addr
: a string passed as the argument to the -header control argument of the dprint command. Some of the valid strings are the same as in XXX_dest above.

admin_online
: User_id of user to receive "crank ran" message.

The value command has two entry points: value$dump and value$set. For details, see the *Multics Administration, Maintenance and Operations Commands* manual, Order No. GB64.

UNDERSTANDING SYS_ADMIN_DATA

The sys_admin_data segment contains registration keywords and an interlock to prevent multiple editing of the same segment. The sys_admin_data segment is located in >udd>SysAdmin>lib.

MODIFYING SYS_ADMIN_DATA

The sys_admin_data segment contains the following items:

- A lock used to prevent more than one administrator at a time from editing the accounting data.
- Character strings identifying the name, address, and telephone of the "User Accounts Office" for use in printing bills.
- Mailing banner data for preparing a cover page for long bill items.
- Default attributes for new projects:

  minimum and maximum ring
  maximum grace
  load control group
  project attributes

The admin_util command (see the *Multics Administration, Maintenance and Operations Commands* manual, Order No. GB64) may be used to display and edit the following values in sys_admin_data:

- admin lock
- user accounts office info
- Mailing banner
- minimum ring
- maximum ring
- maximum grace
- load control group
- project attributes

Each of these values is discussed below.

Admin Lock

One of the values contained in admin_util is a lock that prevents two system administrators from modifying the data bases simultaneously. The administrative lock can be controlled with the following two admin_util control arguments:

lock wait_time

This control argument attempts to lock the lock in sys_admin_data. If the lock is already locked by another process and remains locked for more the the specified wait_time (in seconds), then an error message is printed. The default wait_time is 60 seconds.

unlock

The unlock control argument attempts to unlock the lock in sys_admin_data. If the lock was not locked by the process that is executing the command, an error message is printed. If it was locked by an existing process (other than the one executing the command), it is not unlocked.

User Accounts Office Info

The following control arguments control installation-dependent items that are printed on monthly bills and other administrative documents. The values of these items must be enclosed in quotes if they contain any blanks or other special characters.

user_accts XX

The XX string is the official name of the office responsible for Multics billing, for example: "Fiscal Office" or "Accounting Department." It is the first line of a return address printed on bills by the mailing_page_ subroutine. It may be up to 64 characters long.

user_accts_addr XX

The XX string is the address of the above office, for example, a building and room number or a mail station. It is the second line of a return address printed on bills. It may be up to 64 characters long.

user_accts_phone XX

The XX string is the phone number of the above office. It is also printed on bills. It may be up to 16 characters long.

Mailing Banner

The mail banner can be changed with the control argument b1 XX to b3 XX. The XX strings, each of which may be up to 10 characters long, are printed in large letters by the mailing_page_ subroutine, as a set, below and to the left of the address of the bill recipient. One example of their value is:

```
INTER
DEPARTMENT
MAIL
```

Minimum Ring

The initial ring at which a new or existing project can use is specified by the init_ring control argument of the admin_util command. The ring number specified by must be a single digit from 1 to 7 inclusive. This is the default initial ring for new projects. The default value is 4.

Maximum Ring

The maximum ring of a project is specified by the max_ring control argument of the admin_util command. The ring number specified is the default max_ring for new projects. The default value is 5.

Maximum Grace

The maximum grace of a project can be set with the grace control argument of the admin_util command. The grace time specified by N is the default grace time (in minutes) for a new project. The grace time specified by N is the length of time primary users retain their primary status (protected from preemption). The default value is 2880 minutes (48 hours), which really means "never to be subject to preemption."

Load Control Group

The group XX control argument of the admin_util command identifies the default load control group for new projects. This name may be up to 8 characters long. The default value is "Other".

Project Attributes

The attributes keyword of admin_util lets you select the attributes for a specific project. The attribute string must be acceptable to the parse_attributes_ subroutine, enclosed in quotes if it contains any blanks. This string sets the default attributes for a new project. The anonymous attribute or anon attribute, which allows anonymous users to be registered on a project, can only be assigned by a system administrator. For a more detailed description of registration of anonymous users done by the Project administrator see *Multics Administrators' Manual -- Project* Order No. AK51. The other attributes can be assigned by a project administrator if the system administrator has set them for his project. For convenience, all of the attributes are listed below. The default for an omitted attribute statement is "none." For details on the attributes available, see the admin_util command in the *Multics Administration, Maintenance and Operations Commands* manual, Order No. GB64.

SECTION 41

DELEGATING RESPONSIBILITIES

In addition to the system administrator, a Multics site may also have a project administrator, an accounting administrator or a volume administrator. The functions of these administrators are explained below.

PROJECT ADMINISTRATION

Project administrators are registered Multics users who are in charge of one or more projects. They need not be registered on the projects they administer. A single user may be project administrator for many projects and a project may have up to four administrators. Each project administrator is named in the System Administrator Table (SAT) entry for the project he administers. The project administrator has a copy of the project's Project Master File (PMF) and can use the cv_pmf command to create a PDT from the PMF. The project administrator can also use the install command to install a new or modified Project Definition Table (PDT).

To use the install command, the project administrator must be able to read the proj_admin_seg segment. To place the new copy of a system table to be installed where the system can pick it up, the project administrator must have append permission on the >sc1>update directory.

A *delegated* project has at least one user listed in the SAT entry for the project as the project administrator. If a project is *undelegated* the system administrator or the registration and accounting administrator must perform the project administration function for the project.

It is the system administrator's responsibility to provide the project administrators with access to project directories. The registration and accounting administrator cannot perform this function. Occasionally, the system administrator has to give access to one user's segments to a user on another project.

ACCOUNTING ADMINISTRATION

The registration and accounting responsibilities at Multics installations are designed so they can be delegated to a unsophisticated user, called the registration and accounting administrator. The system administrator can decide if he wants to delegate the registration and accounting responsibilities or carry them out himself.

A registration and accounting administrator (referred to as the accounting administrator) is a registered Multics user in charge of user registration and billing operations. He logs in the same way as other users and, after he logs in, a process is created for him in the same way as for other users. An accounting administrator differs from other Multics users in that:

- An accounting administrator is restricted by a special process overseer to the use of a limited set of administrative commands not available to other users.
- An accounting administrator has access to certain segments to which other users do not.
- An accounting administrator uses special programs to manipulate the accounting data bases. In general, these programs do not make privileged calls; they are ordinary PL/1 programs that manipulate data in ordinary ways.
- The system grants certain requests for an accounting administrator that it does not grant for other users.

The accounting administrator is a user on the SysAdmin project and has a special process overseer:

```
accounts_overseer_
```

This overseer gives the accounting administrator a set of commands designed for accounting administration. Although an accounting administrator has access control privileges that potentially enable him to destroy any segment in the system, he is constrained in his special command system and can perform only a certain specific set of accounting functions.

A system administrator who sometimes logs in as an accounting administrator should be aware that the commands described in the accounts_overseer function differently when called from a process overseer. Instructions for the use of these commands by a system administrator are given in the *Multics Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

The accounting administrator registers new projects and new users on the system monitors the daily, weekly, and monthly accounting jobs and does the billing for the site. An accounting administrator may also be assigned to function as a project administrator for undelegated projects.

VOLUME ADMINISTRATION

A volume administrator is the administrator of a separate logical volume and maintains the quota for the directories that are located on the logical volume. The volume administrator also maintains the access control segments on the logical volume.

For information on system security administrators refer to "Assuring the Security of the File System" in this manual.

SECTION 42

MISCELLANEOUS TAILORING TASKS

TAILORING THE TIME TABLES

All date/time related constants reside in time_info_, such as default format strings, named format strings, day names, month names, offset names, zone names and offsets, and "other" words. All of these are in several languages, one of which is selected as the site default. The time_info_.cds is designed to allow a site to alter the time tables without having to get involved too deeply in how the tables are structured. Below are sample lines from time_info_.cds. Comment lines are included to explain what the code is doing and/or what it means.

Examples

```
1   time_info_: proc;
2
3   dcl (the_language_count init (4),
4        the_zone_count     init (44),
5        the_keyword_count  init (16),
```

First, are three constants which define the size of the site's tables. They must be adjusted if additional languages, time zones, or time format keywords are added to the table.

```
6        english            init (1),
7        french             init (2),
8        german             init (3),
9        spanish            init (4),
10       Default_Language   init (1),
```

There must be a named constant for each language which is in the table. These names must be numbered sequentially beginning at 1. The members of the site's default language is given in the next constant.

```
11        Fill_From          init (1)
12                           fixed bin int static options (constant);
```

Time zone names are defined in a 2-dimensional array, with zone as one dimension and languages as the second dimension. Elements of the array are zone names (the name of a zone in a particular language). convert_date_to_binary_ requires that the entire array be filled (ie, a zone name must be specified for each zone in every language). However, sometimes the name of a zone in a particular language is not known. This problem is resolved by filling in each unspecified zone name with the corresponding name in another language. The Fill_From constant indicates which language should be used to fill such empty array elements. When choosing the Fill_From language, it is important to note that a zone name must be explicitly specified for every zone in the Fill_From language.

```
13        call setup;
```

This initializes internal data structures to indicate "empty" in every field.

```
14        call set_format ("system_date_time",
15         "^my/^dm/^yc//^Hd^99v.9mh/^xxxxza^xxxda");
16        call set_format ("system_date", "^my/^dm/^yc");
17        call set_format ("system_time", "^Hd:^MH");
```

For each of the defined time format keywords, the keyword name and its time format value is given. The keywords for system_date_time, system_date, and system_time define the default site formats for date/time, date and time strings used in many commands. Care must be taken when changing the default site formats to ensure that the new format is reasonably short (so that commands that include a date/time with other data in output will not produce overlength lines) and fixed length (so that output of commands producing columns of data will line up properly).

Note that the date, time, and date_time keywords are not specified here since these keywords identify per-process values while time_info_.cds defines per-system constants.

```
18        call set_language (french, english, "anglais");
```

For each language present, its name in each other language must be present. This call to the routine says:

> In french, the word for english is "anglais".

```
19        call set_month_name (english, Jan, "Jan", "January");
```

For each language present, the month names must be set. This call to the routine says:

> In english, the abbreviation and name for the first month are "Jan" and "January".

There are named constants in the procedure for each month:

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

```
20      call set_day_name (spanish, Mon, "lun", "lunes");
```

For each language present, the day names must be set. This call to the routine says:

In spanish, the abbreviation and name for Monday are "lun" and "lunes".

There are named constants in the procedure for each day:

Mon Tue Wed Thu Fri Sat Sun

```
21      call set_offset (english, Hour, "hr","hours","hour","this");
```

For each language present, the offset names must be set. This call to the routine says:

In english, the abbreviation, plural, and singular for hour are "hr", "hours", and, "hour". The gender of the word "this" that is appropriate for use with the english word hour is "this".

There are named constants in the procedure for each unit:

Year Month Week Day Hour Minute Second Microsecond

The word "this" can be used with the names of offset units to produce time strings. For example:

this_month/1/this_year

defines the date of the first day of the current month. However, the word used for "this" in some languages varies depending upon the gender of the offset unit. For example, in french we have:

```
22      call set_offset (french,Year,"an","ann{es","ann{e","cette");
23      call set_offset (french,Month,"m","mois","mois","ce");
```

Note that national characters such as "{" in "ann{e" are used instead of overstriking to add accent marks to letters. On french terminals, "{" produces "e'" and "|" produces "u^". In spanish, "{" produces "e'" and "}" produces "n~".

```
24      call set_word (german, Before, "vor", "?");
```

For each language present, several additional words must be defined. This call to the routine says:

In german, the word for before is "vor", and the "?" means that there is no known abbreviation for before in german.

There are named constants in the procedure for these words:

```
Before On After Or Noon Midnight
Now Today Yesterday Tomorrow Fiscal_Week
```

```
25        call set_zone (english, "fwt",  "fwt",  +1,
26                  French Winter Time");
27        call set_zone (french,  "fwt",  "hfh",  +1,
28                 "Heure Francais d'Hiver");
29        call set_zone (english, "sast", "sast", +9.5,
30                 "South Australia Standard Time");
```

Time zone names may be defined for many zones in one or more languages. These calls to the routine say:

1. In english, the zone identified as "fwt" is to have an offset of −1 hours (1 hour later than Greenwich Mean Time (GMT)), an abbreviation of "fwt", and name of "French Winter Time".

2. In french, the zone named "fwt" is to have the abbreviation "hfh", and the name "Heure Francais d'Hiver". All settings of a given name must have the same offset.

3. The third example illustrates that a zone abbreviation may have 4 characters in it and that offsets are not necessarily in integral hours.

The second argument in this call is an internal id of a zone. It provides a mechanism to reference a given entry in each language. As released, this is the abbreviated zone name in english. The fourth argument is an offset (in hours) to be added to a time expressed in GMT to produce a time in the named time zone. Time zones west of 0 longitude need negative offsets; those east of 0 longitude need positive offsets. Offset values are real numbers ranging from −11 to +12 hours (i.e., −12 <offset <= +12). Each internal zone_id may be referenced only once per language. Each reference to a given zone_id must specify the same offset value. A given offset value can be associated with many different zone identifiers (e.g., gmt, ut, and z are all zone_ids for an offset of 0 hours).

Note that the default system time zone is not specified in time_info_. Instead, it is given in the CLOK config card, which can be modified in BCE.

```
31        call build;
```

This signals that all data has been filled in.

```
32        return;
33    /* all internal procedures here */
34    end time_info_;
```

After all the data has been presented, the build checks several things. If it finds any that it does not like, it will not generate the table. The following situations cause checks to be made:

1. All tokens (words in a table) must be unique within a language.

2. There may not be any uninitialized fields. If a zone name is not set in some language, the name in the "fill from" language is used. After that, if any field has not been set, it is an error. This includes the error detected when a zone is not define in the Fill_From language.

3. The ambiguity of all the tokens is checked. If a string has one meaning in one language and a different meaning in another, that token will have an ambiguity flag associated with it. If a string is used for the same purpose in several languages, the use of it will be flagged to indicate that it alone cannot determine which language is being handled. These flags are for use by convert_date_to_binary_.

A binary search table is built to aid convert_date_to_binary_ in quickly locating a token among all these words. The details of this internal table are not of importance to a user and are subject to change as convert_date_to_binary_ requires.

After time_info_.cds is changed and compiled, the resultant time_info_ object segment must be updated into the bound_library_1_.archive in the hardcore library. This archive must be bound (via the bind command), and the new bound segment must be placed on a new Multics system tape (via the generate_mst command).

SETTING YOUR SEARCH RULES

The set_system_search_rules command is a highly privileged command used in the initializer process to set a site's default search rules for all processes on the system. For details on the use of the set_system_search rules command, see the Administration, Maintenance and Operations Commands manual.

SECTION 43

SYSTEM MAIL TABLE

The mail table is a table of names for mailing addresses. It enables users to address one another by name alone, without knowing or using the addressee's Project_id. For instance, a user could send mail to "Rickert" without knowing whether Rickert is on the Mktg or Sales project. The mail table can also include names for mailing lists and Forum meetings.

Each Multics system has one mail table. The system administrator creates the initial mail table with the create_mail_table command. Names for the initial mail table are taken from the person name table (PNT). The names used for mailing addresses in the initial table are the Person_ids of users, and the mailing addresses are the mailboxes in users' default project. For instance when "Rickert" is taken from the PNT, the address "Rickert.Sales." is placed in the mail table because "Sales" is the default project in Rickert's PNT entry.

Once the mail table has been created, the system administrator can add to it and update it with the add_mail_table_entry and update_mail_table_entry commands. The delete_mail_table_entry command is used to delete names of users who are no longer registered and of mailing lists and Forum meetings that no longer exist. Also, when PNT entries are added or deleted, corresponding mail table entries are automatically added or deleted. Users can also alter their own mail table entries with the set_mailing_address command (see the Multics Extended Mail System User's Guide, Order No. CH23).

Because the mail table and PNT contain a parallel set of entries, any time the PNT is compacted, the mail table should also be compacted with the compact_mail_table command. When this is done, the old mail table is renamed to mail_table.MM/DD/.NHMM.

SECTION 44

DATA MANAGEMENT ADMINISTRATION

This section provides information to administrators on how to bring up Data Management on Multics. Other aspects of Data Management administration appear throughout this manual under general topics. Also see the section entitled "Multics Data Management" in the *Multics Programmer's Reference Manual*, Order No. AG91, for a complete description of this product.

SITE PREPARATIONS FOR DATA MANAGEMENT

Data Management is part of a Multics software release as of MR11, but is not required to boot Multics. Specific steps must be taken, however, to use Data Management both at new and existing sites. These steps are outlined below and detailed under separate heading. An error condition is signaled for anyone attempting to use Data Management at a Multics site not setup to run Data Management.

1. Data_Management.Daemon Registration and Access
2. Multics Bootload Requirements for Data Management
3. Creating the Data Management Hierarchy
4. Setting Quota for Data Management
5. Shaping the Run-Time Environment
6. Booting the Data Management System

Data_Management.Daemon Registration and Access

As a first step to using Data Management at a new site, the DM daemon (Data_Management.Daemon) must be registered and given access to certain system tables and access control segments. A tool is provided to perform these tasks automatically. When you run >tools>acct_start_up.ec as part of Multics installation, it does the following:

- Registers the special user identity Data_Management on the Daemon project
- Gives this daemon r access to the absentee_user_table, daemon_user_table, and answer_table
- Gives this daemon rw access on bump_user.acs and process_termination_monitor.acs in >sc1>admin_acs
- Sets this daemon's initproc to its own process overseer, dmsd_overseer_, and sets the ^vinitproc and ^multip attributes

Existing Multics sites must perform the above operations manually. See "Special User Identities" in Section 33 for information on registering the daemon. If they do not already exist in >sc1>admin_acs. you have to create the segments bump_user.acs and process_termination_monitor.acs. Also see Section 23 for the appropriate daemon access to privileged gates.

Other DM daemon access settings are as follows:

- s access on >site>Data_Management (see "Creating the Data Management Hierarchy")
- sma access on each AIM directory (see "Sites Running with AIM")
- rw access on dm_configuration (see "Shaping the Run-Time Environment")

Note that >tools>system_start_up.ec includes a sample login line for Data_Management.Daemon that has been commented out. This line is an example; the daemon's message coordinator ID has no required format. The daemon should not be logged in until all data management site preparations have been completed.

Multics Bootload Requirements for Data Management

The dbmj card must be included in the configuration deck at Multics bootload time, if Data Management is to be run at the site. The purpose of this card is to establish the parameters of before journal management for the system, specifically:

- The maximum number of before journals allowed to be open simultaneously
- The maximum number of pages (control intervals) that may be held in memory at any one time
- The maximum number of synchronized segments allowed to have 4K, 16K, 64K, and 256K AST pool entries

Protected DM files consist of synchronized segments, which hold modified pages (control intervals) in main memory until their respective before images are written out to disk. The dbmj card determines the maximum number of active synchronized segments in the system and how many pages can be held in the various sized pools before they have to be flushed to disk.

The maximum-number-of-pages parameter also determines the size of the dm_journal_seg_, a hardcore segment used by all DMS systems at the site to map each modified control interval to its respective before image, and to track the time of modification and when the before image is recorded. The dm_journal_seg_ segment is built at Multics bootload time.

Note that if you boot Multics without the dbmj card and subsequently wish to use Data Management, you will have to bring down the system and bring it back up with the dbmj card in the config deck.

Data Management also requires that the dirlock_writebehind tuning parameter be turned on to ensure that directories (and their trees) containing protected DM files are flushed to disk when updating occurs, to keep the file system intact. While Data Management guarantees the integrity of protected DM files, it has no control over directories containing these files. The DM daemon must be able to locate the containing directories in order to recover files left inconsistent by failures. This parameter ensures that the daemon will always be able to locate these directories, even after a system crash without ESD.

You can enable directory flushing either by including a parm dirw card in the config deck at Multics bootload time, or by using the change_tuning_parameters command after the system is booted. Since there can be overhead associated with this activity, the extent to which Data Management is used should determine how directory flushing is enabled. For example, if Data Management is to be an integral part of operations at the site, you should include the parm dirw card in the config deck. But if Data Management is to be brought up selectively (e.g., for testing), you can enable the dirlock_writebehind tuning parameter with the command on an as-needed basis. Whichever method you use, directory flushing must be enabled to ensure recovery of DM files in the event of failure.

The config deck is described in detail in the *Multics System Maintenance Procedures* manual, Order No. AM81.

Creating the Data Management Hierarchy

To run Data Management, you must create a system directory at >site>Data_Management (with a suggested addname of dm) to contain the per-AIM directories. If your site does not run with AIM, you must then create the system_low directory under >site>Data_Management; this becomes the per-AIM directory for an AIM access class of level zero and no categories. The DM daemon requires sma access and all users of Data Management must have s access to this directory. If the site runs with AIM, other considerations apply as described below under "Sites Running with AIM."

The per-AIM directory (system_low) will contain the DMS per-bootload directory, created automatically at DMS initialization. It will also contain the data management configuration table, the logs directory, and the system default before journal (unless specifically located elsewhere). The configuration table is created and installed as a separate step (see "Shaping the Run-Time Environment" below). The logs directory and default before journal are setup by the DM daemon at DMS initialization.

If your site is not running with AIM, you may wish to use system_low as a link back to >site>Data_Management to save a directory in the hierarchy. In that case, the ACLs mentioned above must be set on >site>Data_Management for the daemon and for all DMS users.

Setting Quota for Data Management

There should be sufficient quota under >site to accommodate the per-AIM directory (system_low). If the site is running with AIM, quota assignments have to be made for each per-AIM directory created (see below). The following figures in pages are representative of quota requirements for a per-AIM directory.

```
system default journal      4000

logs directory              200

DMS configuration table     1

bootload directory          30
```

Note that the system default before journal claims most of the quota required (here the default is taken). Optimal journal size is a site-dependent value, geared to the volume of transactions, their length in time, and the degree of concurrency. Journal quota is requisitioned at the time of creation; it is not dispensed as the journal is used. The logs directory would start at zero and use quota as required, creating new segments in the process. Periodic maintenance of the logs directory is expected.

Shaping the Run-Time Environment (Defining and Creating the DMS Configuration File)

With the per-AIM directory in place, you can shape the DMS run-time environment by creating a configuration source file, converting it into a configuration table segment, and loading it into the directory. It is through the configuration file that you actually establish the parameters of the run-time system, including guidelines for daemon behavior, delays for the various shutdown stages, size and location of the system default before journal, maximum number of processes and transactions allowable, and debugging actions.

The configuration source file is an ASCII segment provided as input to the cv_dmcf command. The pathname of the segment must have the .dmcf suffix. The command converts the source into a configuration table segment with the same entryname and a .dmct suffix. After you have run the command to convert the source file, you must install the table by copying it into the per-AIM directory with the name dm_configuration. The DM daemon must have rw access on dm_configuration. The cv_dmcf command, including instructions on creating the configuration source file, is described in the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64.

Booting the Data Management System

After the DMS configuration table is installed in the per-AIM directory, you can boot DMS by logging in the DM daemon. At this point, you can add the automatic login line for Data_Mangement.Daemon to system_start_up.ec. This line should include the message coordinator id that conforms to your site requirements for routing daemon console output.

SITES RUNNING WITH AIM

If your site runs with multiple AIM access classes, you must take specific steps to setup Data Management at each AIM access class that is to use Data Management. By definition, system_low is the per-AIM directory for an AIM access class of level zero and no categories. For any other AIM access class using Data Management, a per-AIM directory must be created under >site>Data_Management as described below. Also ensure that Data_Management.Daemon has a login range of lowest to highest to encompass all AIM access classes in use, as a DM daemon is required for each per-AIM directory. This will also require that the multip attribute be set in the PMF.

To bring up Data Management at an arbitrary AIM access class, do the following:

1. Login at the access class of >site>Data_Management and execute the command

   do "create_dir [encode_access_class &r1] -acc &r1 -quota &2"

   encode_access_class active function returns the encoded version of the requisite access class denoted by the first argument to the command line. The second argument specifies the quota to be moved from >site>Data_Management to the per-AIM directory being created. Sufficient quota should be allotted as described under "Setting Quota for Data Management."

2. Logout and log back in at the AIM access class of the directory just created and create a DMS configuration table as described for the system_low directory. You may wish to use the same source file, or you may wish to change some source statements, but you must create the table using the cv_dmcf command and copy it into the per-AIM directory just created.

3. Login a DM daemon at the AIM access class for which a per-AIM directory was just created. The daemon then proceeds with DMS initialization, creating the per-bootload and logs directories under the per-AIM directory.

Repeat these steps for each AIM access class to use Data Management.

ADMINISTRATIVE COMMANDS

Commands are available to administrators to setup, monitor, control, and test the data management environment. These commands, all of which are described in the *Administration, Maintenance, and Operations Commands* manual, Order No. GB64, are listed below.

- The cv_dmcf command is used to create the data management configuration table (see "Shaping the Run-Time Environment").
- The dm_lock_status, dm_lock_meters, and before_journal_meters commands are used to monitor system performance (see "Monitoring Data Management Performance").

- The dm_system_shutdown, dm_set_journal_stamps, and dm_send_request commands are used to take specific action in controlling events during operations.

- The dm_set_system_dir command is used to setup a DMS system directory in the user process for test purposes.

Other data management commands, intended for use by DMS users in general, are documented in the *Multics Commands and Active Functions* manual, Order No. AG92. Aspects of some of these commands, notably before_journal_status and transaction, are privileged, requiring access to dm_admin_gate_.

MONITORING DATA MANAGEMENT PERFORMANCE

Facilities are available to administrators to monitor data management performance in order to fine-tune the system. Included among these facilities are administrative commands that provide the following services:

- See what locks are currently held or awaited as a means of improving concurrent access throughput (dm_lock_status command).

- View the frequency of calls to the locking mechanism and how often deadlocks occur, in order to evaluate contention among applications (dm_lock_meters command).

- Check on the overall performance of the before journal manager by analyzing various metering data kept on individual journals and system-wide journal operations (before_journal_meters command).

There are also data management logs that are kept as part of the Multics system logging facility. Information pertaining to DMS initialization, shutdown, and normal operations is available. A separate logs directory is maintained under each per-AIM directory. The data management log location is {system dir} > {per-AIM dir} >logs>dm_system_log. See Section 24 for details on system logs and how to view them.

OPERATIONAL CONSIDERATIONS

Administrators should be aware of certain aspects of data management operations that directly affect users of DMS, particularly with regard to shared use of before journals and backing up DM files. The administrative role in these aspects is discussed below.

Before Journal Storge Limits

The amount of storage a transaction can use in a given journal can be administratively controlled. This capability is useful for shared before journals (e.g., the system default journal) so that individual transactions do not accidentally or maliciously usurp journal space to the detriment of other transactions.

A transaction storage limit can be imposed through the bj_mgr_call command as described in the *Multics Commands and Active Functions* manual, Order No. AG92. For the system default journal, a transaction limit of ten percent of the journal space is acceptable, but system performance at individual sites will dictate how this parameter should be tuned.

Backing Up Files

The Multics dump/retrieve mechanisms have certain limitations with regard to their use in the data management environment.

Requests for volume and hierarchy dumps or retrievals must be submitted from a ring equal to or lower than the target file object. Since DM files are ring 2 file objects, requests cannot be submitted from the user rings, requiring the intervention of administrators to effect this activity.

Volume and hierarchy dumpers do not recognize the atomicity of transactions and therefore may not produce consistent dumps, given the potentially volatile state of the data. Any dumping involving DM files (or other MRDS data bases) should be performed when no updating is taking place.

APPENDIX A

I/O DAEMON TABLES SOURCE LANGUAGE DESCRIPTION

The following statements may appear anywhere within the source file.

Time: <number>;
defines the number of minutes that the coordinator saves a processed request. When segment deletion is requested, it is delayed this amount of time. If some problem is discovered that necessitates the reprocessing of requests, those requests performed less than <number> minutes ago can be restarted. One, and only one, Time statement must appear in the file.

Max_queues: <number>;
defines the default number of priority queues for each request type. The maximum value of <number> is 4. (Queue 1 is the highest priority and queue 4 is the lowest priority.) One, and only one, Max_queues statement must appear in the file.

Line: <name>;
defines the name of a logical line_id and denotes the beginning of a line description. Any subsequent substatements (see below) apply to this line until the next Line, Device, or Request_type statement is encountered. Any <name> may be chosen; it can be a maximum of 32 characters and cannot contain spaces or periods. There may be up to 360 Line statements. This statement is optional.

Device: <name>;
defines the name of a major device and denotes the beginning of a device description. Any subsequent substatements (see below) apply to this device until the next Line, Device, or Request_type statement is encountered. Any <name> may be chosen; it can be a maximum of 24 characters and cannot contain periods or spaces. At least one Device statement must appear in the file.

Request_type: <name>;
defines the name of a request type and denotes the beginning of a request type description. Any subsequent substatements (see below) apply to this request type until the next Line, Request_type, or Device statement is encountered. Any <name> (not containing periods or spaces) may be chosen; it can be a maximum of 24 characters. Currently, however, the names of request types used by the dprint or dpunch commands are restricted to a maximum of eight characters. At least one Request_type statement must appear in the file.

End;
marks the end of the source language description. Unlike all other statements, it has no parameter. Any text occurring beyond the End statement is ignored. One, and only one, End statement must appear in the file.

SUBSTATEMENTS FOR LINES

The following substatements describe various attributes of a line and may appear in any order following a Line statement.

channel: <name>;

defines the name of the communications channel to be attached when using the logical line_id (defined in the Line statement). It is normally a teletype channel identifier for an RJE station. The <name> may be up to 32 characters and cannot contain any spaces. One, and only one, channel substatement must be given for each Line statement.

att_desc: <string>;

defines the attach description to be passed to the remote_teleprinter_I/O module. The <string> may be up to 256 characters and should appear in quotes since there will be embedded spaces. If the control variable ^a appears in <string> it will be replaced by the channel <name> (described above). One, and only one, att_desc substatement must be given for each Line statement.

device: <name>;

defines a major device that can use this logical line_id. At least one device substatement must be given for each Line statement. Any major device specified must also have the line: variable; substatement under the Device statement.

SUBSTATEMENTS FOR DEVICES

The following substatements describe various attributes of a device and may appear in any order following a Device statement.

driver_module: <name>;

defines the name of a procedure to be executed by a driver process when running the associated device. The <name> can be a full pathname or simply an entryname. In the latter case, the search rules are used to locate the procedure. Several standard driver modules are provided by the system (see "Standard Driver Modules" below). One, and only one, driver_module substatement must be given for each Device statement.

default_type: <name>;

defines the default request type for the associated device. The <name> must appear as the parameter in a Request_type statement. Unless overridden by the operator when a driver is initialized, the driver processes requests of this default type.

args: <string>;

defines an argument string to be interpreted by the driver module for the associated device. The <string> may have any arbitrary format up to a maximum of 256 characters. In practice, the composition of the <string> depends on the particular driver module that interprets it. Each driver module has its own conventions for the <string> format (see "Standard Driver Modules" below).

The following three substatements describe alternate methods by which a driver may attach the associated device. These substatements are mutually exclusive. One, and only one, of these substatements must be given for each device statement.

prph: <name>;
names an input/output multiplexer (IOM) peripheral channel through which the associated device can be attached. The <name> must appear on a PRPH card in the configuration deck.

line: <name>;
names a dedicated communications line channel through which the associated device can be attached. If <name> is "variable" the channel can be any logical line_id defined by a Line statement. The driver process must have the dialok attribute in the process definition table (PDT) and the communications channel must be defined as slave in the channel definition table (CDT). See the Communications Reference manual, Order No. CC75, for more information about the PDT and the CDT.

dial_id: <name>;
defines the dial identifier to be used if the associated device is to be dialed to the driver process over a communications line.

The following three substatements describe alternate methods by which the driver may attach a control terminal. These statements are mutually exclusive. If none is specified, the driver assumes that no control terminal is desired.

ctl_line: <name>;
names a dedicated communications line channel through which the control terminal can be attached. The driver process must have the dialok attribute in the process definition table (PDT) and the communications channel must be defined as slave in the channel definition table (CDT). See the Communications Reference manual, Order No. CC75, for more information about the PDT and the CDT.

ctl_dial_id: <name>;
defines the dial identifier to be used if the control terminal is to be dialed to the driver process over a communications line.

ctl_source: <name>;
defines a message coordinator source name to be associated with the driver. (The message coordinator is described in the MOH). A single control terminal accepted by the message coordinator can be used to control many different drivers.

SUBSTATEMENTS FOR REQUEST TYPES

The following substatements describe various attributes of a request type and may appear in any order following a Request_type statement.

accounting: <name>;
defines the name of an accounting procedure to be executed by a driver when processing requests of the associated type. The <name> can be a full pathname or simply an entryname. In the latter case, the search rules are used to locate the procedure. Also, the special <name> "system" can be used to indicate the standard system accounting procedure. If this substatement is omitted, "system" accounting is assumed.

The special <name> "nothing" indicates that no accounting is to be performed. If the site is using multiple rate structures, the daemon process must have read access to the System Administration Table (SAT), unless the special <name> "nothing" was specified.

When the special <name> "nothing" is used, any tail sheet or its equivalent will indicate that there was no charge for the request.

card_charge: "p1, p2, p3, p4";
defines the resource price names for the card_charge of each queue of the request type. This substatement is optional. If it is not specified, the prices from system_info_$io_prices are used. p1 through p4 are resource price names, which are defined using the ed_installation parms command. The prices must be defined before the iod_tables segment is compiled, or the compilation will fail. The price names for each queue must be given in order, from queue 1 to the maximum number of queues for the Request type description. Each price is defined in units of dollars per 1000 cards. Note: card_charge must not be specified if line_charge is specified.

default_queue: <number>;
The default_queue substatement is used to define the default queue for a request type. The value of <number> may be from 1 to max_queues. If not specified, it is set to the value defined in the max_queues substatement, but it will not be greater than 3.

device: <name>;
specifies a device that can be used to process requests of the associated type. The <name> must appear as a parameter in a Device statement. More than one device substatement may be specified for a request type.

driver_userid: <access_name>;
defines the required person and project names for a driver of the associated request type. If omitted, the <access_name> defaults to IO.SysDaemon, which is the standard system driver. Other access names may be used, for example, to provide a project with its own private driver.

generic_type: <name>;
defines the generic type of the associated request type. If the generic type name matches the request type name, then the request type is the default for the generic type. One, and only one, generic_type substatement must be given for each Request_type statement. If the generic type is neither "printer" nor "punch", the list_daemon_requests command must be used with the -brief control argument. The cancel_daemon_requests command cannot be used.

line_charge: "p1, p2, p3, p4";
The line_charge substatement defines the resource price names for the line charge of each queue of the request type. This substatement is optional. If not specified, the prices from system_info_$io_prices will be used. If specified, each price name must be defined in the system price table, or the compilation of the the iod_tables will fail. The price names for each queue must be given in order, from queue 1 to the maximum number of queues for the Request_type description. Each price is defined in units of dollars per 1000 lines.

max_queues: <number>;
The max_queues substatement may be used to define the maximum number of queues for a request type, when it is different from the global Max_queues value. This substatement is optional. The value of <number> may be from 1 to 4.

page_charge: "p1, p2, p3, p4";

The page_charge substatement defines the resource price names for the page charge of each queue of the request type. This substatement is optional. If it is not specified, the prices from system_info_$io_prices are used. If specified, each price name must be defined in the system price table, or the compilation of the iod_tables fails. The price names for each queue must be given in order, from queue 1 to the maximum number of queues for the Request type description. Each price is defined in units of dollars per 1000 pages.

Note: page_charge must not be specified if generic type is "punch".

rqti_seg: <name>;

The rqti_seg substatement is used to define the name of the request type info (rqti) segment to be used with the Request_type statement. This substatement is optional. When specified, <name> must correspond to a segment entryname in the >ddd>idd>rqt_info_segs directory, or the driver will fail initialization. When not specified, no driver will look for an rqti segment for this Request_type statement.

Major devices are defined by the Device statement described earlier. Similarly, minor devices are defined by a minor_device substatement. The minor_device substatement is treated as a substatement for devices and, as such, can be freely intermixed with other substatements for devices.

minor_device: <name>;

defines the name of a minor device belonging to the associated major device and denotes the beginning of a minor device description. Any subsequent substatements (see below) apply to this minor device until the next minor_device substatement or Line, Device, or Request_type statement is encountered. Any <name> may be chosen up to a maximum of 24 characters.

If no minor devices are explicitly defined for a major device, then a default minor device is defined by implication. The primary purpose of a default minor device is to allow certain minor device substatements to be specified for a major device when it has no explicit minor devices. One such substatement is the default_type substatement that was previously described under "Substatements for Devices." In fact, the default_type substatement is actually a substatement for a minor device. When no minor devices are explicitly defined, the default_type substatement applies to the default minor device of the preceding major device. The same is true of all substatements for minor devices.

SUBSTATEMENTS FOR MINOR DEVICES

The substatements below describe attributes of a minor device and may appear in any order following a minor_device substatement or following a Device statement if no minor devices are specified.

minor_args: <string>;

defines an argument string to be interpreted by the driver module for the associated major device. The string may have any arbitrary format up to a maximum of 256 characters. Conventions for the <string> format expected by standard system driver modules are described under "Standard Driver Modules" later in this section.

default_type: <name>;

defines the default request type for the associated minor device. The <name> must appear as a parameter in a Request_type statement.

The device substatement described earlier is a substatement for a request type and is used to name devices that can process requests of a given type. Usually, the parameter of a device substatement is a major device name. However, if minor devices are defined for the major device, then the device substatement parameter must include both the major and minor device names separated by a period (e.g., xyz.printer).

The substatements below describe various attributes of a device class and may appear in any order following a device_class substatement or following a Request_type statement if no device classes are defined.

min_access_class: <access_class>;

defines the minimum access class of a request to be processed in the associated device class. The <access_class> must be a standard access class string as defined by the convert_authorization_ subroutine. If omitted, the default minimum is system_low.

max_access_class: <access_class>;

defines the maximum access class of a request to be processed in the associated device class. The <access_class> must be a standard access class string. If omitted, the default maximum is the access_class string given in min_access_class.

min_banner: <access_class>;

defines the minimum access class banner to be placed on the head sheet of printed output, on the flip cards of punched output, and on the control forms for all output. Normally, the access class of the request is used. However, if this access class is less than that specified for min_banner, then the min_banner value is used. The <access_class> must be a standard access_class string. If omitted, the default min_banner is the access_class string given in min_access_class.

device: <name>;

specifies a device that can be used to process requests of the associated device class. The <name> must appear as the parameter of a Device statement. More than one device substatement may be specified for a device class.

STANDARD DRIVER MODULES

A driver module must be specified for each device defined in the I/O daemon tables. A driver module is a program that embodies specific knowledge of how to manipulate a particular device. The standard driver modules provided by the system are described below.

As mentioned earlier in this section, the <string> argument of the args or minor_args substatements are interpreted by each individual driver module. Even though the format of these strings is defined as arbitrary, each of the standard driver modules support a basic <string> having the following syntax:

```
key= value
```

The key must be unique in <string> and acts like a control argument. The value is the argument associated with the key. Keys and values may not contain commas, but may contain spaces. The key/value pairs are separated from one another by a comma. For example:

```
args:       "dim= device_dim_, form_type= xxx";
```

The complete <string> must appear in quotes and standard Multics quoting conventions apply within <string>. The total length of <string> cannot exceed 256 characters.

The following paragraphs describe the args and minor_args keys that are supported by each of the standard driver modules, as well as other attributes of the I/O daemon tables device specification.

printer_driver_ Module

This driver module should be specified for standard Multics printers. The prph substatement must be specified for the associated device. Multiple minor devices are not supported and the minor_args substatement is ignored. For standard printer operation, no args substatement need be specified. However, the args substatement can be used to define a nonstandard device interface module (DIM) and/or a nonstandard control terminal accountability form type. This is done by including the following key-value pairs in the args substatement.

```
dim= <DIM_name>
```

The "dim=" key defines <DIM_name> to be the DIM through which the device is attached. The default DIM for printer_driver_ is prtdim_.

```
form_type= <form_name>
```

The "form_type=" key defines <form_name> to be the control form type. If not specified, a default control form type is used.

punch_driver_ Module

This driver module should be specified for standard Multics punches. The prph substatement must be specified for the associated device. Multiple minor devices are not supported and the minor_args substatement is ignored. For standard punch operation, no args substatement need be specified. However, punch_driver_ accepts an args statement of the same form as printer_driver_. The default DIM is cpz.

reader_driver_ Module

This driver should be specified for standard Multics card readers. The prph substatement must be specified for the associated device. Multiple minor devices are not supported and the minor_args substatement is ignored. For standard reader operation, no args substatement need be specified. However, the following key-value pairs may be specified in the args substatement:

```
dim= <DIM_name>
```

The "dim=" key defines <DIM_name> to be the DIM through which the device is attached. The default dim for reader_driver_ is crz.

```
station= <Station_id>
```

The "station=" key defines <Station_id> to be the name of the card input station to be associated with this card reader. The default station id is "reader". For example:

```
Device:          reader;
driver_module:   reader_driver_;
prph:            rdra;
default_type:    dummy;

Request_type:    dummy;
generic_type:    dummy;
max_queues:      1;
device:          reader;
```

While the reader_driver_ does not process requests from the coordinator, the syntax of the iod_tables requires the presence of a request_type substatement. This should be a dummy request type to which no users have access to submit requests, as for the reader minor device of the remote_driver_.

Sites with CCU (combined card unit) devices should define two devices: one with punch_driver_ for the punch, and one with reader_driver_ for the reader.

spool_driver_ Module

This driver module should be specified for a major device that will be used to write user print requests onto tape instead of the printer. The prph substatement must be specified, but the <name> need not be an IOM channel. (It is used as an I/O switch name for the tape attachment.) The default type may be omitted if the operator is required to specify the request type each time the "device" is used. For example:

```
Device:          spooler;
driver_module:   spool_driver_;
prph:            tape;
```

The spool_driver_ ignores all args substatements. It does not accept multiple minor devices and does not accept any control terminal specifications.

remote_driver_ Module

This driver module should be specified for all remote printer/punch/reader stations. Two types of stations are supported by the remote driver. A Type I station can be initialized from any one of several communications lines. A Type II station, which does not have an input device, is initialized on a dedicated communications line as a predefined station. The two station types are described separately below because the iod_tables description of each is different.

The driver process must have the dialok attribute in the PDT, and it must have rw access to the access control segment (ACS) of the communications line it will attach. The remote_driver_ can handle one minor device for a card reader and an arbitrary number of minor devices for printers and punches within the limits of the physical remote device and the line protocol. (A minor device for the reader must be specified if the remote device is to read card input.)

The remote_driver_ is designed for maximum flexibility, so its description is rather complex. You should be thoroughly familiar with the iod_tables and Part VII of this manual, "Managing I/O Daemons", and the iod_tables description in the *Multics Maintenance Procedures* manual, Order No. AM81, before attempting to configure a remote driver.

APPENDIX B

AUDIT TABLES AND INCLUDE FILES

The information in this appendix includes:

- the contents of the segment access_operations_.alm in table format (Table B-1). All standard audit messages derive their text from access_operations_.alm
- the text of the <added_info> string for channel/dial-id messages described in Section 25
- and the include files that describe the format of the detailed operation field in the interpreted (-interpret) binary data for standard access audit messages.

The following table illustrates the contents of the segment access_operations_.alm which contains the text for all standard audit messages.

Table B-1. access_operations_.alm Table

TEXT	OBJECT TYPE	OPERATION TYPE	ENTRY NAME
accepting wakeups over a message segment	OTHER	MODIFY_ACCESS	mseg_accept_wakeups
access violation fault - mode	FS_OBJ	UNKNOWN	fault_acv_mode
access violation fault - ring	FS_OBJ	UNKWNOWN	fault_acv_ring
acquisition of rcp object	RCP	MODIFY_ACCESS	rcp_acquire
addition of pnt entry	ADMIN	MODIFY	pnt_entry_add
addition of a message to a message segment	OTHER	MODIFY	mseg_add_message
admin read of pnt entry	ADMIN	READ	pnt_entry_admin_read
allocation of DM journal	OTHER	MODIFY_ACCESS	dm_journal_allocate

Table B-1 (cont.). access_operations_.alm Table

```
TEXT

      OBJECT TYPE      OPERATION TYPE    ENTRY NAME

assign resource for writing
      RCP              MODIFY            rcp_assign_write

assign resource for reading
      RCP              READ              rcp_assign_read

attachment of logical volume
      SPECIAL          MODIFY_ACCESS     attach_lv

cancel rcp object reservation
      RCP              READ              rcp_cancel

closing a message segment
      OTHER            READ              mseg_close

compacting a message segment
      OTHER            MODIFY            mseg_compact

copy rcp meters or rcp data
      RCP              READ              rcp_copy_info

counting messages in a message segment
      OTHER            READ              mseg_get_count

creating a message segment
      OTHER            MODIFY            mseg_create

creation and acquisition rcp of object
      RCP              MODIFY_ACCESS     rcp_register_acquire

creation of fs_obj
      FS_OBJ_ATTR      MODIFY            fs_obj_create

creation of rcp object
      RCP              MODIFY            rcp_add_device

creation of rcp registry
      RCP              MODIFY            rcp_copy_registry

creation of rcp object
      RCP              MODIFY            rcp_register
```

Table B-1 (cont.). access_operations_.alm Table

TEXT	OBJECT TYPE	OPERATION TYPE	ENTRY NAME
deleting a message from a message segment	OTHER	MODIFY	mseg_delete_message
deleting a message segment	OTHER	MODIFY	mseg_delete
deletion of pnt entry	ADMIN	MODIFY	pnt_entry_delete
deletion of fs_obj	FS_OBJ_ATTR	MODIFY	fs_obj_delete
deletion of rcp object	RCP	MODIFY	rcp_delete_device
deletion of registry	RCP	MODIFY	rcp_delete_registry
deletion of rcp object	RCP	MODIFY	rcp_deregister
detachment of logical volume	SPECIAL	MODIFY_ACCESS	detach_lv
excessive segment state changes	FS_OBJ_ATTR	UNKNOWN	excessive_seg_state_chg
freeing of DM journal	OTHER	MODIFY_ACCESS	dm_journal_free
illegal procedure fault	NONE	UNKNOWN	fault_ipr
initiation of fs_obj	FS_OBJ_ATTR	READ	fs_obj_initiate
interprocess wakeup	SPECIAL	MODIFY	ipc_wakeup
listing of rcp resources	RCP	READ	rcp_list

Table B-1 (cont.). access_operations_.alm Table

```
TEXT

      OBJECT TYPE      OPERATION TYPE    ENTRY NAME

login read of pnt entry
      ADMIN            READ              pnt_entry_login_read

manual clear rcp object contents
      RCP              MODIFY            rcp_clear

modification of pnt entry
      ADMIN            MODIFY            pnt_entry_modify

modification of fs_obj
      FS_OBJ           MODIFY            fs_obj_contents_mod

modification of fs_obj attribute prop
      FS_OBJ_ATTR      MODIFY            fs_obj_attr_mod

modification of fs_obj status prop
      FS_OBJ_ATTR      MODIFY            fs_obj_status_mod

modification of printed access labels
      FS_OBJ_ATTR       MODIFY_ACCESS    io_daemon_set_page_labels

modification of fs_obj access
      FS_OBJ_ATTR      MODIFY_ACCESS     fs_obj_access_mod

modification of fs_obj access class
      FS_OBJ_ATTR      MODIFY_ACCESS     fs_obj_reclassify

modification of security out-of-service
      FS_OBJ_ATTR      MODIFY_ACCESS     fs_obj_set_soos

modification of process audit flags
      NONE             UNKNOWN           process_audit_flags_modify

modification of system audit flags and thresholds
      NONE             UNKNOWN           system_audit_thresh_modify

modification of system AIM privilege
      NONE             UNKNOWN           system_privilege_modify
```

Table B-1 (cont.). access_operations_.alm Table

TEXT

OBJECT TYPE	OPERATION TYPE	ENTRY NAME
modification of rcp registry		
RCP	MODIFY	rcp_update_registry_header
modify rcp object atributes		
RCP	MODIFY	rcp_set
modify rcp object access attributes		
RCP	MODIFY_ACCESS	rcp_set_access
modifying message segment attributes		
OTHER	MODIFY	mseg_attr_mod
modifying message segment access information		
OTHER	MODIFY_ACCESS	mseg_access_mod
network read of pnt entry		
ADMIN	READ	pnt_entry_network_read
opening a message segment		
OTHER	READ	mseg_open
overflow of logical volume		
NONE	UNKNOWN	log_vol_full
preload of rcp object		
RCP	READ	rcp_preload
priv read of pnt entry		
ADMIN	READ	pnt_entry_priv_read
read of fs_obj		
FS_OBJ	READ	fs_obj_contents_read
read of fs_obj prop		
FS_OBJ_ATTR	READ	fs_obj_prop_read

Table B-1 (cont.). access_operations_.alm Table

TEXT	OBJECT TYPE	OPERATION TYPE	ENTRY NAME
read rcp accounting info	RCP	READ	rcp_account
read the rcp error count	RCP	READ	rcp_error_count
reading message segment access information	OTHER	READ	mseg_access_read
reading message segment attributes	OTHER	READ	mseg_attr_read
reading a message in a message segment	OTHER	READ	mseg_read_message
reading and deleting a message from a message segment	OTHER	MODIFY	mseg_read_delete_message
reading of DM journal attributes	OTHER	READ	dm_journal_read_attr
reconstruction of rcp registry	RCP	MODIFY	rcp_reconstruct_registry
release rcp object acquisition	RCP	MODIFY_ACCESS	rcp_release
reservation of rcp object	RCP	READ	rcp_reserve
sending a wakeup over a message segment	OTHER	MODIFY	mseg_wakeup_send
status of rcp object	RCP	READ	rcp_status
termination of fs_obj	FS_OBJ_ATTR	READ	fs_obj_terminate
unassignment of rcp object	RCP	READ	rcp_unassign

Table B-1 (cont.). access_operations_.aim Table

TEXT		
OBJECT TYPE	OPERATION TYPE	ENTRY NAME
unload of resource		
RCP	READ	rcp_unload
updating a message in a message segment		
OTHER	MODIFY	mseg_update_message
writing of DM journal attributes		
OTHER	WRITE	dm_journal_write_attr

The following table contains the <added_info> string text for the channel/dial-id messages described in Section 25.

Table B-2. <Added info> string for Channel/Dial-id Messages

<Added info> can contain:
<an interpreted system status code>
<a logout reason string (e.g. logout, autolog, hangup, bump, new_proc, cpulimit, termsgnl, badsignl, lhbrief, etc)>
<the name of the channel involved>
<an illegal channel starname>
without listen
without hangup
using comm privilege
hangup
error
release
dialgrab
dialnang
Access check failed after slave dialup
tandd_attach control order failed
registered {privileged}
privileged
dialid released {(keeping N dialed consoles)}
rw access is required to >sc1>rcp>tandd.acs
dialid shutoff
Null channel name specified
Channel <channel_name> not found
Channel <channel_name> does not exist
Channel not attached to this process
requested authorization is greater than maximum authorization
set_required_access_class failed
dial_out control order failed
Invalid channel specified
Channel not currently dialed out
Channel not currently dialed out for this process
error in dialing out
process lacks access to registered dial acs for <dial-id>
Dial id not in use by this process
No dial id in use by this process
another process is registered dial server for <dial-id>
process lacks comm priv for privileged service of <dial-id>
process lacks comm priv for privileged dial-out
process already serving <dial-id> {(registered)}, cant serve <dial-id2>
User lacks dialok attribute

DETAILED OPERATION FIELD OF STANDARD BINARY HEADER

The following include files describe the format of the detailed operation field in the interpreted binary data for standard access audit messages. These apply only to audit messages which describe RCP resources and hardcore file-system objects.

```
/*  BEGIN include rcp_ops.incl.pl1  */

dcl       1 detailed_operation   unaligned based (ops_ptr),
            2 given,
              3 potential_attributes
                                 bit (1),
              3 desired_attributes
                                 bit (1),
              3 potential_aim_range
                                 bit (1),
              3 aim_range        bit (1),
              3 owner            bit (1),
              3 acs_path         bit (1),
              3 location         bit (1),
              3 comment          bit (1),
              3 charge_type      bit (1),
              3 usage_lock       bit (1),
              3 release_lock     bit (1),
              3 user_alloc       bit (1),
            2 priv_gate_call     bit (1),
            2 search             bit (1),
            2 force              bit (1),
            2 process            bit (1),
            2 owner              bit (1),
            2 pad                bit (1);

/*  BEGIN include rcp_ops.incl.pl1  */


/* BEGIN include file fs_obj_access_codes.incl.pl1 */

/* Major file system operations. */

dcl  access_operations_$fs_obj_create bit (36) aligned ext;
dcl  access_operations_$fs_obj_delete bit (36) aligned ext;
dcl  access_operations_$fs_obj_initiate bit (36) aligned ext;
dcl  access_operations_$fs_obj_terminate bit (36) aligned ext;
dcl  access_operations_$fs_obj_contents_read bit (36) aligned ext;
dcl  access_operations_$fs_obj_contents_mod bit (36) aligned ext;
dcl  access_operations_$fs_obj_prop_read bit (36) aligned ext;
dcl  access_operations_$fs_obj_attr_mod bit (36) aligned ext;
dcl  access_operations_$fs_obj_status_mod bit (36) aligned ext;
dcl  access_operations_$fs_obj_access_mod bit (36) aligned ext;
```

```
/* Detailed operations. */

dcl  FS_OBJ_CONNECT fixed bin (18) uns init (1) static options (constant);
dcl  FS_OBJ_BC_MOD fixed bin (18) uns init (2) static options (constant);
dcl  FS_OBJ_TRUNCATE fixed bin (18) uns init (3) static options (constant);
dcl  FS_OBJ_ACL_MOD fixed bin (18) uns init (4) static options (constant);
dcl  FS_OBJ_RING_MOD fixed bin (18) uns init (5) static options (constant);
dcl  FS_OBJ_ACL_RING_MOD fixed bin (18) uns init (6) static options (constant);
dcl  FS_OBJ_RENAME fixed bin (18) uns init (7) static options (constant);
dcl  FS_OBJ_COPY_SW_MOD fixed bin (18) uns init (8) static options (constant);
dcl  FS_OBJ_DAMAGED_SW_MOD fixed bin (18) uns init (9) static options (constant);
dcl  FS_OBJ_DNZP_MOD fixed bin (18) uns init (10) static options (constant);
dcl  FS_OBJ_ENTRY_BOUND_MOD fixed bin (18) uns init (11) static options (constant);
dcl  FS_OBJ_MAX_LEN_MOD fixed bin (18) uns init (12) static options (constant);
dcl  FS_OBJ_SAFETY_SW_MOD fixed bin (18) uns init (13) static options (constant);
dcl  FS_OBJ_SYNC_SW_MOD fixed bin (18) uns init (14) static options (constant);
dcl  FS_OBJ_VOL_DUMP_SW_MOD fixed bin (18) uns init (15) static options (constant);
dcl  FS_OBJ_AUTHOR_MOD fixed bin (18) uns init (16) static options (constant);
dcl  FS_OBJ_BC_AUTHOR_MOD fixed bin (18) uns init (17) static options (constant);
dcl  FS_OBJ_BACKUP_TIMES_MOD fixed bin (18) uns init (18) static options (constant);
dcl  FS_OBJ_DATES_MOD fixed bin (18) uns init (19) static options (constant);
dcl  FS_OBJ_DT_DUMPED_MOD fixed bin (18) uns init (20) static options (constant);
dcl  FS_OBJ_FOR_RELOADER_MOD fixed bin (18) uns init (21) static options (constant);
dcl  FS_OBJ_SONS_LVID_MOD fixed bin (18) uns init (22) static options (constant);
dcl  FS_OBJ_SOOS_MOD fixed bin (18) uns init (23) static options (constant);
dcl  FS_OBJ_MOVE_QUOTA fixed bin (18) uns init (24) static options (constant);
dcl  FS_OBJ_CORRECT_QUSED fixed bin (18) uns init (25) static options (constant);
dcl  FS_OBJ_DIR_SALVAGE fixed bin (18) uns init (26) static options (constant);
dcl  FS_OBJ_MDIR_QUOTA_MOD fixed bin (18) uns init (27) static options (constant);
dcl  FS_OBJ_QUOTA_MOD fixed bin (18) uns init (28) static options (constant);
dcl  FS_OBJ_QUOTA_RELOAD fixed bin (18) uns init (29) static options (constant);
dcl  FS_OBJ_RECLASSIFY fixed bin (18) uns init (30) static options (constant);
dcl  FS_OBJ_RECLASSIFY_NODE fixed bin (18) uns init (31) static options (constant);
dcl  FS_OBJ_SEG_MOVE fixed bin (18) uns init (32) static options (constant);
dcl  FS_OBJ_TRP_MOD fixed bin (18) uns init (33) static options (constant);
dcl  FS_OBJ_VOLUME_RETRIEVE fixed bin (18) uns init (34) static options (constant);
dcl  FS_OBJ_IACL_MOD fixed bin (18) uns init (35) static options (constant);
dcl  FS_OBJ_CREATE_BRANCH fixed bin (18) uns init (36) static options (constant);

/* END include file fs_obj_access_codes.incl.pl1 */
```

RCP DETAILED OPERATION FLAGS

The following is a list of the names and meanings of the RCP detailed operation flags:

potential_attributes
: "1"b if the user has specified potential attributes during registration of a resource.

desired_attributes
"1"b if the user has requested the use of a resource by attributes rather than by name or uid.

aim_range
"1"b if the user has specified the aim range during registration, or a set operation on a resource.

owner
"1"b if the user has specified the owner during registration or acquisition of a resource.

acs_path
"1"b if the user has specified the acs_path in a registration, acquisition, or set operation on a resource.

location
"1"b if the location has been specified by the user during a registration, acquisition or set operation of a resource.

comment
same as location.

charge_type
same as location.

usage_lock
same as location.

release_lock
same as location.

user_alloc
same as location.

priv_gate_call
if this operation was made through one of the rcp privileged gates (rcp_sys_, rcp_admin, or rcp_priv_).

search
"1"b if the operation is searching for a resource to meet criteria specified by the user. For example, if the user requests a resource by attributes, then RCP must first search for a resource with the specified attributes. When the search flag is on then auditing is not performed until an appropriate resource is found.

force
"1"b if the -force flag has been used such as during release of of a resource.

process
"1"b if the process has terminated and all resources assigned, reserved or attached to the process must be unassigned, cancelled, or detached by the system.

owner
"1"b if the user is the resource owner.

FILE SYSTEM DETAILED OPERATION CODES

The access_operations_ entries which refer to file system modification are too general for practical use (they note object property modification or directory contents modification). Detailed operation codes are provided to give more specific information about the operation (e.g. which property is modified). There are detailed codes associated with the following access_operations_ entries:

fs_obj_status_mod
fs_obj_access_mod
fs_obj_attr_mod
fs_obj_contents_mod

The codes are described below in numerical order (as in the include file). The access_operations_ entry with which each is associated is shown in parenthesis.

FS_OBJ_CONNECT (fs_obj_contents_mod) (or fs_obj_contents_read)
appears upon a call side or fault side attempt to make a segment known. The access_operations_ entry appearing in the audit will depend upon whether the user has read or write access to the segment. This is the only detailed code which refers to an operation on the contents of a segment. Also, this is the only detailed code which may appear with a "read" type opertion.

FS_OBJ_BC_MOD (fs_obj_contents_mod)
appears upon an attempt to change the bit count of a segment or directory.

FS_OBJ_TRUNCATE (fs_obj_contents_mod)
appears upon an attempt to truncate a segment.

FS_OBJ_ACL_MOD (fs_obj_access_mod)
appears upon an attempt to change the access list of a segment or directory.

FS_OBJ_RING_MOD (fs_obj_access_mod)
appears upon an attempt to change the ring brackets of a segment or directory.

FS_OBJ_ACL_RING_MOD (fs_obj_access_mod)
appears upon an attempt to change both the access list and the ring brackets of a segment or directory in a single hardcore operation.

FS_OBJ_RENAME (fs_obj_status_mod)
appears upon an attempt to change names of a file system entry.

FS_OBJ_COPY_SW_MOD (fs_obj_status_mod)
appears upon an attempt to change the copy switch of a segment or directory.

FS_OBJ_DAMAGED_SW_MOD (fs_obj_status_mod)
appears upon an attempt to change the damaged switch of a segment or directory.

FS_OBJ_DNZP_MOD (fs_obj_attr_mod)
appears upon an attempt to change the dont-null-zero-pages attribute of a segment.

FS_OBJ_ENTRY_BOUND_MOD (fs_obj_status_mod)
appears upon an attempt to set the entry bound of a segment.

FS_OBJ_MAX_LEN_MOD (fs_obj_status_mod)
appears upon an attempt to set the maximum length of a segment.

FS_OBJ_SAFETY_SW_MOD (fs_obj_status_mod)
appears upon an attempt to change the safety switch on a segment.

FS_OBJ_SYNC_SW_MOD (fs_obj_status_mod)
appears upon an attempt to change the synchronized switch on a segment.

FS_OBJ_VOL_DUMP_SW_MOD (fs_obj_status_mod)
appears upon an attempt to change the volume dump control switches on a segment or directory.

FS_OBJ_AUTHOR_MOD (fs_obj_status_mod)
appears upon an attempt to change the author of a segment or directory.

FS_OBJ_BC_AUTHOR_MOD (fs_obj_status_mod)
appears upon an attempt to change the bit count author of a segment or directory.

FS_OBJ_BACKUP_TIMES_MOD (fs_obj_status_mod)
appears upon an attempt to set the backed up time status of a segment or directory (privileged).

FS_OBJ_DATES_MOD (fs_obj_status_mod) appears upon an attempt to set the date status of a segment or directory (privileged).

FS_OBJ_DT_DUMPED_MOD (fs_obj_status_mod)
appears upon an attempt to set the backed up time status of a segment or directory (privileged).

FS_OBJ_FOR_RELOADER_MOD (fs_obj_status_mod)
appears upon an attempt to set a number of object status properties when reloading the object (privileged).

FS_OBJ_SONS_LVID_MOD (fs_obj_status_mod)
appears upon an attempt to set the sons logical volume of a directory (privileged).

FS_OBJ_SOOS_MOD (fs_obj_status_mod)
appears upon an attempt to change the security-out-of-service flag for a segment or hierarchy subtree (privileged).

FS_OBJ_MOVE_QUOTA (fs_obj_contents_mod)
appears upon an attempt to move segment or directory quota between two directories.

FS_OBJ_CORRECT_QUSED (fs_obj_contents_mod)
appears upon an attempt to correct the quota used in a directory (privileged).

FS_OBJ_DIR_SALVAGE (fs_obj_contents_mod)
appears upon an attempt to salvage a directory.

FS_OBJ_MDIR_QUOTA_MOD (fs_obj_contents_mod)
appears upon an attempt to set the segment quota on a master directory.

FS_OBJ_QUOTA_MOD (fs_obj_contents_mod)
appears upon an attempt to set the segment or directory quota in a directory without affecting its parent's quota (privileged).

FS_OBJ_QUOTA_RELOAD (fs_obj_contents_mod)
appears upon an attempt to restor segment or directory quota upon a hierarchy reload (privileged).

FS_OBJ_RECLASSIFY (fs_obj_access_mod)
appears upon an attempt to reclassify a segment or directory (privileged).

FS_OBJ_RECLASSIFY_NODE (fs_obj_access_mod)
appears upon an attempt to reclassify a hierarchy subtree (privileged). vacating a physical volume (privileged).

FS_OBJ_TRP_MOD (fs_obj_contents_mod)
appears upon an attempt to set the time-record-product in a directory (privileged).

FS_OBJ_VOLUME_RETRIEVE (fs_obj_contents_mod)
appears upon an attempt to copy the contents of a segment or directory during a volume retrieval (privileged).

FS_OBJ_IACL_MOD (fs_obj_contents_mod)
appears upon an attempt to change the initial access list of a directory.

FS_OBJ_CREATE_BRANCH (fs_obj_contents_mod)
appears upon an attempt to add a new branch or link entry to a directory.

INDEX

S

T

HONEYWELL BULL
Technical Publications Remarks Form

TITLE: MULTICS SYSTEM ADMINISTRATION PROCEDURES

ORDER NO. AK50-03B

DATED DECEMBER 1987

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

PLEASE FILL IN COMPLETE ADDRESS BELOW.

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

CUT ALONG LINE

CUT ALONG LINE

FOLD ALONG LINE

PLEASE FOLD AND TAPE–
NOTE: U.S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 39531 WALTHAM, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Honeywell Bull Inc.
200 Smith Street MS486
P.O. Box 9199
Waltham, Massachusetts, 02254-9832

FOLD ALONG LINE

Honeywell Bull

MULTICS

SYSTEM ADMINISTRATION PROCEDURES ADDENDUM B

SUBJECT

Changes to the Manual

SPECIAL INSTRUCTIONS

This is the second addendum to AK50-03, dated May 1985, and its associated addendum, Addendum A, dated July 1985. Insert the attached pages into the manual according to the collating instructions on the back of this cover. See "Significant Changes" in the Preface for a description of the new material.

Note:

Insert this cover after the manual cover to indicate the updating of the document with Addendum B.

SOFTWARE SUPPORTED

Multics Software Release 12.1

ORDER NUMBER

AK50-03B

December 1987

51305
1088
Printed in U.S.A.

Honeywell Bull

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
Front cover, blank	Front cover, blank
Title page, Preface	Title page, Preface
iii through xvii, blank	iii through xvii, blank
4-1 through 4-4	4-1 through 4-5, blank
7-1 through 7-3, blank	7-1 through 7-3, blank
8-3, 8-4	8-3, 8-4
18-11 through 18-14	18-11, 18-12
	18-13, blank
	18-13.1, 18-14
23-5, 23-6	23-5, 23-6
24-5, 24-6	24-5 through 24-8
25-7, 25-8	25-7, 25-8
25-19 through 25-36	25-19 through 25-28
	25-29, 25-29.1
	25-29.2, 25-30
	25-31 through 25-36
33-3 through 33-6	33-3, 33-4
	33-5, 33-5.1
	33-5.2, 33-6
i-1 through i-17, blank	i-1 through i-11, blank
Blank, rear cover	Blank, rear cover

File No.: 1L13

AK50-00B

MULTICS

SYSTEM ADMINISTRATION PROCEDURES ADDENDUM A

SUBJECT

Additions and Changes to the Manual

SPECIAL INSTRUCTIONS

This is the first addendum to AK50-03, dated May 1985. Insert the attached pages into the manual according to the collating instructions on the back of this cover. See "Significant Changes" in the Preface for a description of the new material.

Note:
Insert this cover after the manual cover to indicate the updating of the document with Addendum A.

SOFTWARE SUPPORTED

Multics Software Release 11.0

ORDER NUMBER

AK50-03A

July 1985

44088
7.5C1085
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
iii, iv	iii,iv
ix through xvii, blank	ix through xvii
25-1 through 25-14	25-1 through 25-36
26-1 through 26-9, blank	26-1 through 26-11, blank
	B-1 through B-14
i-1 through i-17, blank	i-1 through i-17, blank

7/85

 File No.: 1L13 AK50-03A

Honeywell Bull

Corporate Headquarters:
3800 West 80th St., Minneapolis, MN 55431
U.S.A.: 200 Smith Street, MS 486, Waltham, MA 02154
Mexico: Hamburgo No. 64, Col. Juarez Delegacion Cuauhtemoc, 06600 Mexico, D.F.
U.K.: Great West Rd., Brentford, Middlesex TW8 9DH, England **Italy**: 32 Via Pirelli, 20124 Milano
Canada: 155 Gordon Baker Road, North York, Ontario M2H 3P9 **New Zealand**: 14/16 Liverpool Street, Auckland 1
Asia: 4/F, Shui on Centre, 6-8 Harbour Rd., Wanchai, Hong Kong **Australia**: 124 Walker Street, North Sydney, N.S.W. 2060

42899, 1088, Printed in U.S.A.

AK50-03