HEWLETT PACKARD

RTE-A

System Manager's Manual

Software Services and Technology Division
11000 Wolfe Road
Cupertino, CA 95014-9804

Manual Part No. 92077-90056
E0495

Printed in U.S.A. April 1995
Fifth Edition

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

First Edition	Aug 1987	Rev. 5000 (Software Update 5.0)
Second Edition	Jan 1989	Rev. 5010 (Software Update 5.1)
Update 1	July 1990	Rev. 5020 (Software Update 5.2)
Third Edition	Dec 1992	Rev. 6000 (Software Update 6.0)
Fourth Edition	Nov 1993	Rev. 6100 (Software Update 6.1)
Fifth Edition	Apr 1995	Rev. 6200 (Software Update 6.2)

Preface

The *RTE-A System Manager's Manual* describes the duties of a system manager. It is designed to complement the *RTE-A System Generation and Installation Manual*.

The manual gives a brief overview of system management and refers to the other manuals in the RTE-A manual set for detailed information on specific elements of the system. Particularly useful are the *RTE-A System Design Manual*, the *RTE-A Backup and Disk Formatting Utilities Manual*, the *RTE-A Driver Reference Manual*, and *RTE-A Programmer's Reference Manual* for maintenance and expansion.

The manual also describes resource management in the multiuser environment through the Group and User Management Program (GRUMP) utility, the Security/1000 utility of VC+, and the SECTL utility. Note that information on system security tables and the Security/1000 library (SECLIB.1000) are contained in this manual so that the system manager can control the distribution of this information.

Conventions Used in this Manual

The following conventions are used in this manual:

Convention	**Meaning**
Uppercase letters	In command syntax, uppercase letters indicate characters that must be entered as shown.
Lowercase italic letters	In command syntax, lowercase italic letters indicate user-supplied information.
[ ]	In command syntax, brackets indicate optional items.
	In examples of interactive sessions, brackets indicate default values.
, or blank	A comma or blank are command delimiters.

Table of Contents

List of Illustrations

Tables

System Management Overview

The RTE-A Operating System (HP product number 92077A) is a powerful and flexible program providing an environment that allows concurrent user access to system resources. As system manager, your duties include:

- Designing and Planning – determining the system requirements and structure; deciding whether or not to use the Primary System; installing the Primary System if it is to be used.
- Generating and Installing – creating a customized system from the Primary System if there is no existing system; getting the new system running.
- Maintaining – supporting the existing system's operation and integrity by regenerating as required; performing backup of the operating system; performing backup of user files and applications; keeping the system current with software updates; answering questions about system operation; keeping software and hardware documentation current and available.
- Recovering and Shutting Down System– restoring system operation from the backup system copies made at installation if needed; performing system shutdown as needed.

The process of system management is described by the flowchart in Figure 1-1.

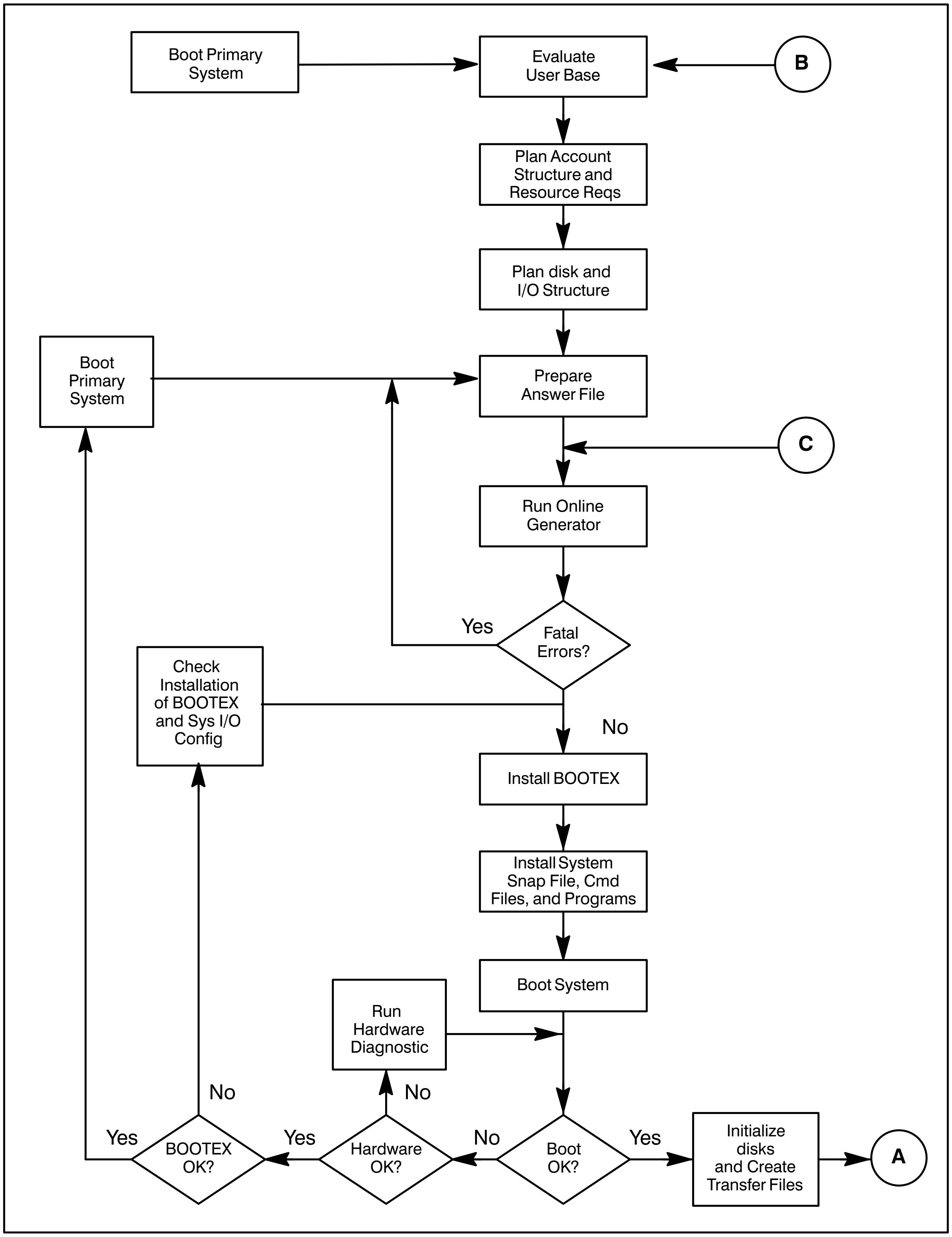

Figure 1-1. System Management Procedural Overview

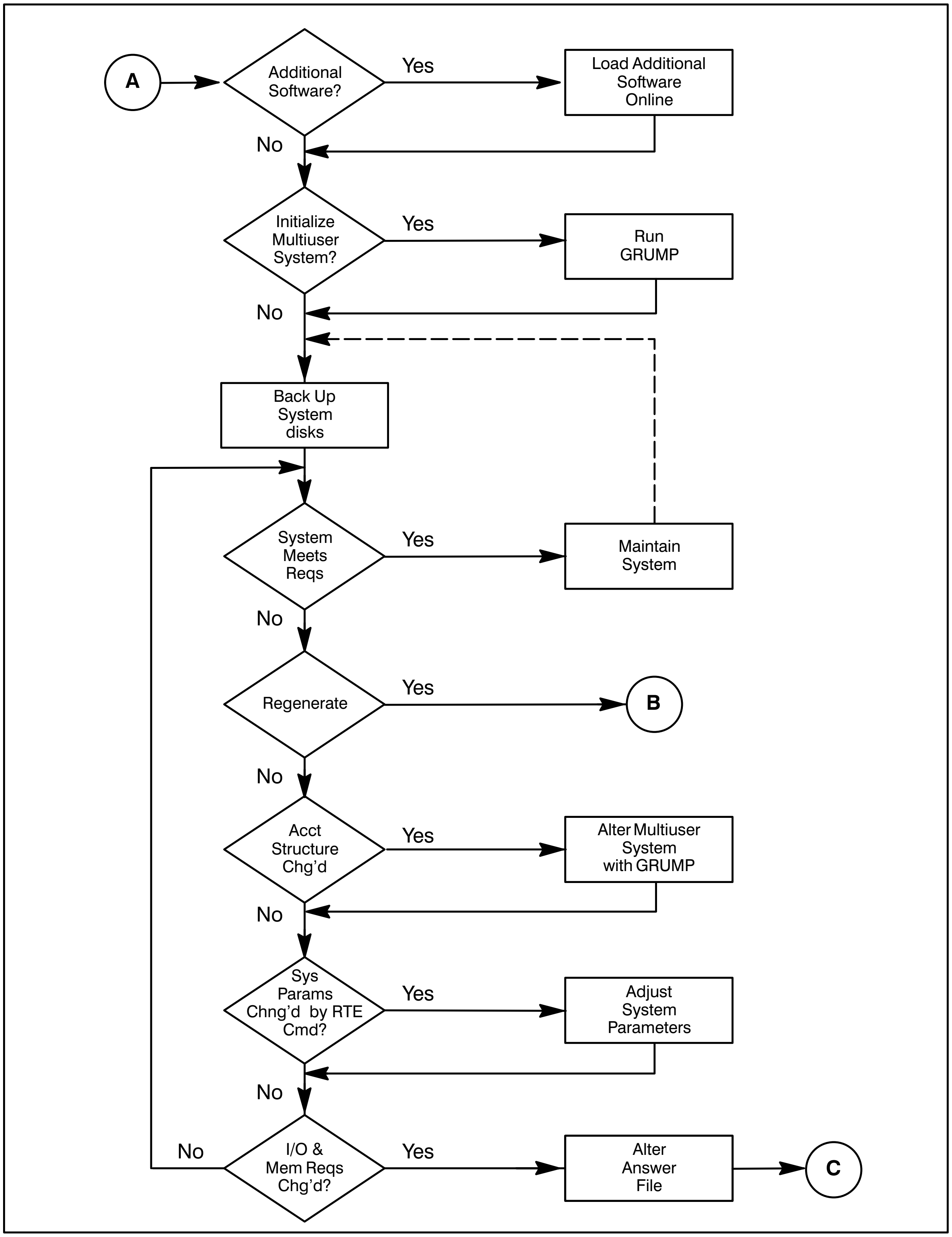

Figure 1-1. System Management Procedural Overview (continued)

Designing and Planning

In planning system requirements, system managers need to know:

- Who will be using the system
- What applications will be run
- What system resources and peripherals will be required

You must consider types of applications to be run, required system resources and peripherals, and possible future expansion. Refer to the *RTE-A System Design Manual*, part number 92077-90013, for detailed information on system design concepts, software planning, and I/O design and planning.

The steps in system planning are:

- Plan the account structure, and the disk volume and cartridge requirements.
- Plan the computer I/O structure, determining select codes for each card, LU numbers for each device, and the drivers required to control these cards and devices. See the *RTE-A System Generation and Installation Manual*, part number 92077-90034, for details.
- Plan the RTE-A memory allocation by determining the number of reserved partitions, size of SAM (System Available Memory), size of XSAM (Extended SAM), number of ID segments, class numbers, and resource numbers.

Figure 1-2 provides a sample form for organizing your system planning. Modify it to meet your specific needs.

Determining User Requirements

To determine user requirements, you should create a questionnaire and use it to interview potential system users. Most users do not think in terms of disk tracks, memory or disk-resident programs, or priority levels when describing their needs. Therefore, the questions should be readily understood and provide the kind of information that can be translated into data useful for system generation, initialization, and maintenance.

The topics any questionnaire should cover include:

- User Categories.
- Applications.
- Peripheral Resource Usage.

USER CATEGORIES

_____ General Users

_____ Operators

_____ Application Programmers

_____ System Programmers/System Managers

Suggested User Categories

Set Categories Appropriate to Your System

# of concurrent users _________

User Names

____________________ ____________________ ____________________

SYSTEM APPLICATIONS

Special Program Requirements

EMA Programs

Size ___ Should EMA be shareable? Y N With what programs? _________

# of class Numbers ______________________________________

# of words of SAM and XSAM ______________________________________

# of Resource Numbers ______________________________________

# of shared programs ______________________________________

Number of programs active at one time ______________________________________

Labeled Common Size ______________________________________

Unlabeled Common Size ______________________________________

Program partition size ____________________ Partition reserved? Y N

PERIPHERAL RESOURCE USAGE

Subsystems ______________________________________

_____ Using hierarchical file system

_____ Using FMGR file system

_____ Common Data Base/File Access

_____ Line Printer Access

_____ Cartridge Tape Drive Access

_____ Magnetic Tape Unit Access

_____ Others

_____ Special Requirements

Suggested Peripherals

Set Categories Appropriate to Your System

Figure 1-2. Determining User Requirements

User Categories

The first step in system planning is to identify the levels of user sophistication on the system. This section is primarily applicable to systems with VC+ (Virtual Code Plus). VC+ is a Hewlett-Packard software product (HP product number 92078A) that provides multiuser capability, and extensive directory, file, and command access control. Four user categories are defined below, but they should be customized for your system.

General users are at the lowest level of user sophistication. Users in this group interface with the system only to the extent of operating specific programs or command files. No programming knowledge is necessary, and very little knowledge of the system is required. Users are expected to follow predefined procedures when working with the computer.

Operators are at the next level. Users at this level may require knowledge of the editor and cursory knowledge of the file system. Only limited access to the system functions is needed.

Application programmers are at the fourth level. Most RTE-A users fall into this category. These users have knowledge of, for example, operator commands and programming calls. They are expected to take advantage of most system capabilities including operation of compilers, management of data bases, manipulation of the file system, performance of network operations, and so forth. However, these users are not concerned with the activities of the other users on the system. Detailed system knowledge is not usually required.

System programmers and system managers are at the highest level of sophistication. These users have a good working knowledge of system operation. They are capable of changing overall system operating parameters.

The user categories information can be the basis for establishing groups. You should also determine the number of users who will be on the system at the same time (concurrent users). Refer to Chapter 2 for information on the multiuser account system.

System Applications

This section deals with intended system applications. The answers to the questions presented in this section determine how system resources are allocated and how to set up system parameters.

All information such as the class number totals or program partition sizes should be collected for each individual application, and tabulated for the total system requirement. If system size allows, extra resources should be added to the total user requirement to accommodate future applications.

The answers collected will determine:

- Subsystems required—Select the HP-supplied subsystems, languages, utilities, and user application programs to be used on the system. Determine which subsystems (for example, NS or IMAGE) will be used because all of them have individual system requirements. Encourage users to consider future expansion as well as current needs.

- Response time requirements—Users should be asked their terminal and real-time response requirements. Based on their responses, modules may be given higher priority levels and/or assigned to partitions. For example, response considerations in a real-time environment may dictate that certain programs be memory-resident at all times. These programs may need to have priorities higher than the priority of the real-time fence. Also, you will use the information about these programs to determine sizes of reserved partitions.

- Memory requirements—You must have information about real-time programs when creating a boot command file so that you can create reserved partitions large enough to accommodate those programs. Partitions must be large enough to accommodate any large application programs users will run. Use the number of reserved partitions and user programs to adjust the number of the memory descriptors specified in the generation file. EMA (Extended Memory Area) usage is another factor to be considered. User applications using the EMA feature require partitions of a certain minimum size. This affects the amount of physical memory required in the system. Therefore, users should be asked for the maximum EMA space used by their application programs, and whether or not the information in the EMA is to be shared by more than one program.

Peripheral Resource Usage

Suggested peripheral resources are shown in Figure 1-2. You should tailor the list of peripherals to your own system. Each user, or group of users, must provide the following information:

- Will the user be storing files or creating data bases on the system? If so, how many and how big? Does the user require disk space on a permanent or temporary basis? This indicates the amount of disk space (if any) to allocate to the user.

- Will the user's files be accessed by other users in the system? Will this user access other users' files? Which users? Does this user have files that cannot be shared? These questions are important in systems using VC+ because file access can be restricted to the individual user, made available to all system users, or made available to members of a group.

- Does the group require a special peripheral? For example, a peripheral may be necessary for one group's application. Another group on the same system, but involved in a different application, may have the use of that same peripheral restricted.

- Will the users need EDIT, the text editor, for program development or other text processing? If so, adequate disk space should be made available to accommodate the scratch files created by EDIT. You can dedicate a disk volume to storing temporary editor scratch files. Refer to the *EDIT/1000 User's Manual*, part number 92074-90001, for editor loading information. You must also make disk space available for other programs such as LINK (the linkage editor), FST (File Storage to Tape), and TF (Tape Filer) that require large amounts of space for scratch files.

- Does the disk have an integrated Cartridge Tape Drive (CTD)? If so, reserve a buffer area on the disk for the CTD. This area is referred to as the disk cache. The size of the disk cache for the CTD operations greatly improves the data transfer rate. Refer to the *RTE-A System Generation and Installation Manual,* part number 92077-90034, for information on how to reserve this area.

After determining the devices that will be used by your applications, assign them LUs (Logical Units). Integrated devices have special requirements because each device needs a separate LU number (and in some cases, a separate driver). Some applications require assignment to LUs that are numbered less than 64. (There is a maximum of 255 LUs.) Conventions in assigning LUs to logical devices are:

- System console = LU 1.
- Line printer = LU 6.
- Streaming magnetic tape = LU 7.
- Magnetic tape = LU 8.
- CS/80 cartridge tape = LU 24.
- Disk LUs = LUs up to and including 63.
- Maximum LU number = 255.

Determine if there are special needs such as time requirements or access restrictions. For example, one group may need to use the magnetic tape drive for several hours each night; another group may need the same tape drive early in the morning, and still another group may need the tape drive to be available most of the day. Coordinating user needs is very important when determining peripheral requirements.

Disk Management Considerations

In RTE-A, disks and files are managed by two file system interfaces, CI (Command Interpreter) and FMGR, the file manager. In CI, the preferred interface, files and directories are managed and protected by read/write protection. The CI hierarchical file system divides the disk into large areas of free blocks. These areas are identified by LU numbers and are called disk volumes. Files in each disk volume are managed by directories and subdirectories that maintain information on the files.

The FMGR file system divides a disk into fixed-size cartridges that are identified with cartridge reference numbers (CRN). The CRN can be any two-character alphanumeric string. Each cartridge has a cartridge directory containing pertinent information on all files stored on that cartridge. Files cannot be organized into directories and subdirectories. Although CI is the preferred file management system, it is useful to have at least one FMGR cartridge in case some utility requires one.

Table 1-1 shows a comparison of the two file systems. The CI file system lets you assign several users to one large disk volume, making the free space on the volume available to all users. Usually, each user is given a private set of hierarchical directories for file management. As shown in Table 1-1, CI provides time stamps for file creation, last update and last access; it provides file unpurge capabilities, and file names up to l6 characters long. FMGR, on the other hand, provides only one directory per disk LU, and the files on each LU must have unique names limited to six characters.

Table 1-1. RTE-A File System Comparison

	CI File System	FMGR File System
File name	1–16 characters	1–6 characters
Cartridge/ directory	1–16 characters in directory	1–2 characters or numeric cartridge names
File Security	Protection based on directory and file ownership	Security code used for file protection
FileTypes	File type extensions describing the contents of the files	Defines the structure of the files
File Mask	Mask qualifier and special characters in file name	Limited file masking
File Size	Extendable	Extendable
Time Stamps	Create, access, and update times handled by the file system	None
Subdirectory	Subdirectories within directories and other directories	None
File recovery	Operator recoverable immediately after purge	None
Spooling	Can be done interactively and programmatically	Can be done interactively and programmatically
Incremental Backup	Done in conjunction with TF and FST utilities	None

File Volume

A volume is a self-contained section of a disk. Each volume is independent of any other volume. Directories and files can never cross volumes. Each physical disk drive consists of one or more volumes. Volumes can never cross physical drives. Each volume is identified by a logical unit (LU) number. Volumes are always identified by their disk LU number. Volume LU numbers range from 1 to 63 inclusive.

Each volume contains information about its layout. This information includes the names of all the global directories in the volume, as well as a table that indicates which disk blocks have been allocated in the volume. This table is called a bit map because the table is composed of bits rather than addresses or values.

When designing a disk layout, be aware of the advantages and disadvantages of using a single LU, versus several large LUs. If you use a single LU, you will benefit from initial low maintenance, and provide for some applications, such as a data base file, that require a large unit. However, a single LU per disk tends to waste disk space. Using several large LUs has the advantages of better space utilization, being easier to pack, and being easier to work with. In general, it is better to divide a disk into several large disk LUs. Make sure that your scratch file and swap file are on large LUs for system operation purposes.

Some advantages of larger LUs are:

- There is more room for files and subdirectories under one global directory.
- The disk does not need packing as often.
- Space for very large files, such as scratch and swap files, is readily available.

Some advantages of smaller LUs are:

- The backup of an individual LU is faster.
- Required FMGR LUs can be allocated a smaller amount of disk space.
- Users can be restricted to specific LUs. The advantage here is that you can use Security/1000 to restrict users to specific LUs, and volume ownership can be specified.

Mounting a volume makes that volume, and all the directories and files on it, available to the operating system. Dismounting a volume removes that volume, and makes the directories and files on it inaccessible to the system. These operations are performed infrequently; except with removable media such as floppy disks where disks must be mounted after they are installed and dismounted before they are removed.

Mounting a volume will initialize it if there are no valid data on the volume. Initializing a volume sets up information needed by the operating system including the list of directories and the bit map for keeping track of space use. If the disk volume does not have a valid FMP (File Management Package) or FMGR directory, you are prompted of this before the volume is initialized. This prevents accidentally corrupting volumes that are not FMP or FMGR types (for example, backup utility volumes). If the disk volume does have a valid FMP or FMGR directory, the volume is initialized.

Except for duplicate directory names on two or more volumes, the order in which disk volumes are mounted is unimportant. If a global directory on the newly-mounted disk volume has the same name as a previously mounted global directory, the new directory is inaccessible.

The CI MC (Mount) command does not place reserved blocks at the beginning of the volume. If reserved blocks are required, use the CI IN (Initialize) command. For more information on CI file volumes, refer to the *RTE-A User's Manual*, part number 92077-90002.

Directory Organization

Directories are the central CI file system data structure. Directories maintain the file system status. All information pertaining to a file is maintained in a directory.

Directories may be included in other directories, these are considered subdirectories. Nesting of subdirectories is allowed to provide a hierarchical file system structure. At the top is a root directory that contains only unique global directories. There is one root directory per disk volume. Mounting a disk volume makes directories on that volume accessible.

Global directory names must be unique in the file system. An abbreviated form of the directory name is kept in free space in D.RTR; each global directory requires five words. For this reason, a limit of approximately 300 global directories is recommended. This limit applies only to global directories and not to subdirectories. Therefore, convert global directories into subdirectories with the CI MO (Move) command if the limit is reached.

Generating and Installing

If your system is already running, refer to the “Maintaining” section.

If you are installing a system for the first time, the *RTE-A Primary System Software Installation Manual*, part number 92077-90038, which comes with the software, provides the procedure for installing a Primary System.

The Primary System is a tested, factory-configured operating system that can be booted up immediately after installation. It provides a working starter system that can be used to test the functions of the installed hardware. It can be used by the system manager to regenerate a customized system. It can also be used by all levels of users to gain familiarity with the RTE-A operating system features.

Each Primary System tape includes all the RTE-A operating system components needed to generate a new system. The Software Numbering File (HP product number 92077A) is a text file that lists all the files included in the RTE-A product directory /RTE-A.

Once your Primary System is running, the steps to install your disk-based system are:

1. Prepare the boot LU by creating a BOOTEX area, if necessary.

   BOOTEX is a memory-based system that has the sole responsibility for loading and initializing a disk-based operating system.

2. Install BOOTEX, if necessary.

3. Prepare the boot command file.

   The boot command file contains commands for the BOOTEX system to initialize the disk-based system.

4. Install system, snap, and boot command files on the boot LU.

5. Create the required directories and program files.

6. Set up the startup program and the Welcome file.

   The Welcome file is a CI command file that is run by the RTE-A Operating System to mount volumes, initialize devices, and so forth.

7. Boot and initialize the system.

8. Verify operation and back up the system.

Generation

System generation consists of preparing a system generation answer file and running RTAGN, the RTE-A online generator. An answer file contains prepared responses to generator program questions when building the new system. Normally, an existing answer file (a sample answer file is shipped with the RTE-A operating system) is edited to create a new answer file. See the *RTE-A System Generation and Installation Manual*, part number 92077-90034, for a sample of an answer file. Appendix A in this manual shows a portion of a sample answer file used in generating a system with Security/1000.

The RTAGN program produces the system, snapshot and list files required to define the system to be installed. The system file contains a memory image of the operating system. At system bootup, the system file is copied from disk or other bootable media to physical memory and executed.

The snap (snapshot) file contains the value of system entry points, system library names, and other system information such as system checksums and the system common checksum. This is used by LINK to load programs online. The current snap file must be copied to the system directory and named SNAP.SNP; LINK automatically searches for that name.

The list file documents what is in the system and where the modules are located. It contains:

- The runstring used to schedule RTAGN.
- The system time at the beginning of the generation.
- A listing of the input commands and comments.
- A list of what was generated into the system and where the different parts of the operating system will be located.
- The location and explanation of any errors that may occur.

You should record and store new generation files and any changed generation files.

Refer to the *RTE-A System Generation and Installation Manual* for instructions on system generation.

Installation

System installation consists of:

- Generating a new system file.
- Preparing the target system hardware and media for boot.
- Booting the new system.
- Setting up a primary program.
- Establishing the account structure, spooling, and directories (if VC+ is used).
- Backing up the new system.

Specific procedures are given in the *RTE-A System Generation and Installation Manual*.

Maintaining

You are responsible for maintaining the integrity of the running system and ensuring that it runs at optimal performance.

You, the system manager, must be prepared to:

- Alter the multiuser account system as required.
- Add features for new applications (for example, security and reserved memory).
- Back up and restore disk LUs and files.
- Alter system parameters to meet new user requirements or change generation parameters.
- Add online software (for example, subsystems such as graphics).
- Keep system and device documents current and available.
- Answer user questions about system operation.
- Interface with Hewlett-Packard in problem reporting, resolution, and system updates (note that this function depends on the level of support a customer has purchased from HP).

If the changes warrant, the system may require a new generation. You will need to regenerate an existing system to:

- Add devices.
- Modify tag size.
- Change the number of ID segments or class numbers.

- Add subsystems.
- Add or delete relocatables.
- Update the system with software revisions.

Many maintenance tasks (for example, altering timeouts) can be done interactively. Ways to make a system more efficient or update it without regenerating include using the maintenance utilities on disks, files, and the accounting system; and performing the tasks listed in the fine tuning section below.

Accounting System

A system with multiple users and sessions is referred to as a multiuser account system. The GRoup and User Management Program (GRUMP) is used to create and maintain the multiuser account system. The GRUMP utility is presented in the "Multiuser Account System" chapter (Chapter 2), which also describes how to plan a system using groups. Use a questionnaire, such as the one in Figure 1-2, as a method of gathering the information needed for multiuser account planning.

Fine Tuning

To get optimal performance, you should:

- Minimize table space and base page links.
- Check device priority and location, making sure that important I/O devices are given higher priority than less important ones and that fast and slow devices are not connected to the same interface card. High-speed devices include disk drives and tape drives. All other I/O devices are considered slow devices.
- Determine the number of ID segments and class numbers that will control the number of system users without hindering those users' activities on the system.
- Run PROMT with the −1 option in the Welcome file to initialize LOGON and CM (VC+ only).
- RP frequently used programs, such as CI and EDIT, with the D option to create prototype IDs for them. Note that to force the system to use the prototype ID, the program must be specified without a directory path when run (VC+ only).
- If you have free memory, generate in a RAM disk. Put frequently used programs on the RAM disk at bootup (in the Welcome file) and RP them with the D option. This way the programs can be dispatched without disk accesses.
- Put the /SCRATCH directory on the RAM disk. This improves the performance of all programs that use the /SCRATCH directory for their processing. This includes MACRO, FTN7X, and PASCAL. It also includes VMA programs such as LINK. If you use the RAM disk for VMA, you may want to consider reducing the working set size as the penalty is much less severe. EDIT's use of the RAM disk for its scratch file is discouraged, as a power fail or crash loses the scratch file and thus the ability to enter EDIT's recovery mode. For this reason it is recommended that a /SCRATCHEDIT global directory be created if you put /SCRATCH on the RAM disk. (See the EDIT manual discussion of /SCRATCH.)

- If you are using the $VISUAL command editing modes in CI (VC+ only), you should RP the CMPLT program in your Welcome file. CMPLT should also be run without wait (XQ, CMPLT) from your Welcome file.

The disk utilities FORMC (CS/80 disk formatting) and FORMF (floppy disk formatting utility) are used to verify the integrity of disks, spare out bad areas of disks, and format and initialize disks. These utilities are described in the *RTE-A Backup and Disk Formatting Utilities Manual*, part number 92077-90249. The file utilities FPACK (File System Pack), FREES (Report Disk Free Space), FOWN (Report File Space by Owner), FVERI (File System Verification), FSCON (File System Conversion), and MPACK (displays and improves disk file allocation) report on the status of disk volumes and are used to manage usable space within the volumes. The FMGR PK command is used to pack FMGR cartridges, reclaiming space previously allocated to files that have been purged. These utilities are described in the *RTE-A User's Manual*, part number 92077-90002.

System Usability

You can use command files to set up or change special environments that make the system easier to use and enhance user productivity. Global and user logon files, for example, are ways of enhancing system usability. A global logon file can be set up on /USERS to list a logon message or do other initialization, and then transfer to the user logon file. User logon files can be set up on the user's working directory to set up UDSPs (User-definable Directory Search Path), run a mail and phone program, and set up commonly used CI variables. You can initiate the logon file when specifying the user's startup program in the GRoup and User Management Program (GRUMP). An example is:

```
 Enter the startup command [xxxx]: ru ci.run::programs /users/logon.cmd
```

where *xxxx* is the default value. Appendix B contains sample logon files.

System Backup

Backups are a method of saving data on a medium other than the current disk (LU). Backup media includes DDS (DAT) tapes, CS/80 cartridge tapes, magnetic tapes, floppy disks, or another disk.

Doing system backup at initialization, and periodically thereafter, protects you and users from losing work and time in the event of unforeseen circumstances such as system shutdown. If system disks are corrupted or destroyed, you can recover by restoring a backup copy of the system.

Backing up your system can be done in two ways:

- Physical backup that saves a physical snapshot of the disk and its exact contents.

- Logical backup that looks for directory and file information on the disk and saves it as directories and files.

The comparison in Table 1-2 on the following page describes the differences between the two types of backup.

Table 1-2. Two Types of Backup

	Physical Backup	Logical Backup
Purpose	Save and restore disk LU on entire disk unit; save exact data on disk – does not have to be files (example, BOOTEX on a CI volume)	Save and restore files or groups of files only
Online/Offline	Online or Offline (utility dependent)	Online only
Backup that can be used to boot system	Yes; usual format for Primary Systems	Not applicable
Utilities/Commands	ASAVE/ARSTR, DSAVE/DRSTR, !PBV, COPYL	FST, TF, FC, LIF, and CO commands in CI; ST and DU commands in FMGR
Advantages	Faster method if LU is full	Faster method if there are only a few files to be saved
	Restore entire LU	Restore on a file and directory basis

System Backup Strategy

The steps in system backup are:

1. Build a memory-based version of your own system.

   After your system is running, build a primary system specifically tailored to your particular requirements (based on responses to a questionnaire similar to the one in Figure 1-2). Build a memory-based version of your system being sure to include D.RTR and ARSTR (or the restore utility corresponding to your physical backup utility). See the *RTE-A System Generation and Installation Manual* for details in building a primary system and creating a memory-based version of the system. Copy the system to tape (see the *RTE-A User's Manual* for details on copying to tape).

2. Make a physical backup of the system and other important LUs.

   Back up all files and programs necessary for minimal operation. Include the `/programs` and `/system` directories as well as programs necessary to bring other programs and files from your physical and logical backups.

3. Make a logical backup.

4. Make periodic file backups using the FST utility.

The *RTE-A Backup and Disk Formatting Utilities Manual* describes the physical backup and restore utilities and the logical backup utilities.

Primary/Physical System Backup

A physical backup is a copy of your system on a bootable medium or a medium that can be downloaded to the system disk with an offline utility. You must have some form of physical backup, or copy of your boot disk LU, in case errors on the LUs with programs, or the swap file, result in system failure. This backup copy can be used to bring up a system quickly.

A physical backup stored on a disk is a bootable system. A physical backup stored on a magnetic tape or a CTD has to be downloaded to a disk before it can be booted. This can be done in the following way:

> Boot a memory-based system that contains the corresponding physical restore utility. Then restore the necessary LUs from tape to disk.

You must prepare a bootable memory-based system that contains ARSTR, the physical restore utility, by using your backup restore utility. This system must be available on some medium other than your system disk. This memory-based system is part of the physical backup.

The physical backup must be able to restore enough of the system to boot up, run the startup program, and bring other programs and files onto the system.

Physical backups can also be used to save other disk LUs besides the boot LU. They can be used to restore bad LUs in situations where your system is running and you can run ARSTR (or the restore utility corresponding to your backup utility) online. In these situations, the backup must be relatively current. You can use incremental logical backups to restore files that changed between the physical backup and the time that errors occurred on the LUs.

It is important that you perform a physical backup each time you install a new revision of the operating system. Older versions of a physical backup may not work with the current operation system.

Physical backups should contain:

- Bootable BOOTEX.
- System file.
- Snap file.
- Boot command file.
- Welcome file.
- System libraries.
- Swap file.
- Necessary directories such as /PROGRAMS and /SYSTEM.

- Necessary programs such as:

CI	Command Interpeter.
CIX	CI Auxiliary Program.
FMGR	File Manager.
D.RTR	Directory Manager.
D.ERR	FMP Error Message Expander, General D.RTR Auxiliary Program.
EDIT	Editor.
DL	Directory List.
LI	List Files.
IO	Display I/O Configuration and Status.
WH	System Status Reporting.
TF	Tape Filer.
FST	File Storage to Tape.
FSTP	FST Auxiliary Program.
LINK	Linker.
PROMT	Prompt Program for VC+.
LOGON	Logon Program for VC+.

- Message catalogs such as:

>TF000	Tape Filer.
>LK000	Linker.
>FS000	File Storage.

- Restore utility corresponding to your physical backup utility.

Physical backups do not save the file structure. They save the physical image of data on the disk. The source and destination LUs must have the same physical characteristics. They must have the same track size and blocks per track. However, the total number of tracks in a disk LU need not match.

Logical Backup

A logical backup saves data on the disk on a per file basis. File structure and attributes are saved.

The FST utility is used to back up and restore files for both CI volumes and FMGR cartridges (except type 0 files) on magnetic tape, DDS tape, or CS/80 cartridges. The TF and FC utilities are also used for backup and recovery but are not as fast as FST. Like FST, TF backs up both CI and FMGR files. FC only backs up FMGR files. You can do a full backup that makes a backup of all files, or you can do an incremental backup of only the CI files by appending delta backups (backups of all files changed since the last backup) to the same tape as the full backup. The next full backup starts on a new tape. The advantages of doing incremental backups are:

- Higher system availability because average delta backup time is faster.
- Less tape used on the average.
- Fewer tapes used because backups can be appended to the same tape.
- Multiple versions of files can be accessed more conveniently for archiving.

The disadvantages of incremental backups are:

- Files take longer to restore.
- The procedures require more effort to understand.
- Incremental backups are not applicable to FMGR files.

CI and FMGR commands are for disk-to-disk backup. Commands used in backups include the CI CO (copy) command and the FMGR ST (store) and DU (dump) commands. Although these CI and FMGR commands are available for disk-to-tape and tape-to-disk, use of FST or TF is more common.

The LIF (Logical Interchange Format) utility provides file interchange between an HP 1000 system and other HP computer systems.

Keys to a Successful System Backup

The following are recommendations for implementing a successful backup scheme:

- Maintain a strict backup schedule.
- Keep users from accessing their files during system backup.
- Only one person should clear the backup bit for a given set of files.
- Label the tapes with date, time, contents, and type of backup.
- Store backups in a safe place.
- Keep system time accurate to maintain correct time stamps.
- Use transfer files for backup and restoration to avoid errors.
- Always verify backups and restores.

Recovering and Shutting Down the System

Common situations in disk-based systems that require system recovery are system crashes, disk failure, or unsuccessful system installation. If you still have a bootable system in any of these situations, you can simply reboot. However, if you do not have a bootable system, you must boot from the Primary System stored on tape. To recover a system backed up with ASAVE, use the ARSTR utility.

Once the system is running, use the FST, TF, and FC utilities to restore files and directories backed up with those utilities. With FST and TF, directories are restored to their original LU when possible. Directories restored by the superuser are owned by their original owner. Directories restored by a non-superuser are owned by the user doing the restore.

Situations may arise that require you to shut down the system. For example, you might shut down the system for hardware maintenance or new card installation. Adverse weather conditions require system shutdown because disks are susceptible to the power surges that may result from electrical storms. If you need to shut down the system, use the procedure recommended in the system installation and service manual.

You can turn off the CPU power when necessary without destroying data on the hard disk. Main memory is lost unless your system is equipped with battery backup.

If you must power down your hard disk, power it down separately after CPU power is off following the instructions accompanying the disk. Refer to the operator manuals for all devices and follow the individual shutdown instructions.

Running Out of SAM

If your RTE-A/VC+ system runs out of SAM (System Available Memory), the PROMT program takes action to allow the system to be recovered. If you get error messages indicating this problem exists, you should perform the following recovery procedures.

If it is possible for PROMT to do so, PROMT disables the multiuser system and allows access to RTE (RTE: prompt is displayed). PROMT attempts to give access to RTE from LU 1. If this is not possible, for example LU 1 is locked, an error message is displayed, and then RTE is made available at the terminal that caused PROMT to be invoked. While the multiuser system is disabled, PROMT does not allow any user access to LOGON, CM, or the SYSTEM> prompt.

When the RTE: prompt is displayed, you can take action to correct the SAM problem (use the OF command to terminate a program using SAM). After you have released some SAM, enter GO RESTR to restore the system to a normal multiuser environment.

Depending on the source of the SAM problem, SAM may be recovered without your help; for example, the programs using SAM may release some SAM. If this happens, you still have to enter GO RESTR to restore the system to a normal multiuser environment. PROMT continues to display the out of SAM message until RESTR has finished executing.

If your system runs out of SAM, it is possible that terminals might get locked to CM or LOGON. If you detect this (you can use the DS (Device Status) command from RTE:) simply OF CM or OF LOGON. You can do this before or after you have restored the system.

If RESTR is not RPed, PROMT attempts to RP RESTR when it detects that the system is out of SAM; however, the RP may fail because of the SAM problem. If RESTR is not RPed and PROMT is unable to RP RESTR, PROMT takes no further action for recovering SAM. The multiuser system is not disabled and RTE is not made available. This is done so that the system can still be restored later.

A preventive step you can take that allows your system to recover if it runs out of SAM is to RP the SAM, WH, and RESTR utilities in your Welcome file at boot time. See *RTE-A Virtual Code+ Manual*, part number 92078-90001, for more details on this. SAM and WH are useful for determining which programs, in which sessions, are causing the problem.

Caution If you run out of SAM and you recover with the above procedure, your system integrity cannot be guaranteed. When this occurs, your only option is to shutdown your applications and reboot your system.

2

Multiuser Account System

Multiuser capability is provided through VC+ (HP product number 92078A). You create and maintain the multiuser account system with the GRoup and User Manager Program (GRUMP) described in detail in Chapter 3.

To prepare for setting up the multiuser account system, you must perform the following tasks:

- Organize the system users into a hierarchy of groups and users. Groups should include sets of users with common characteristics and/or requirements such as members of a project team or individuals with similar functions. Users can be members of more than one group.
- Estimate the number and size of CI file volumes and FMGR disk cartridges in the system. This will depend on your account structure, user application requirements, and the degree of file independence required by various users of the system.

This information can be gathered from interviews with your users. See the “Designing and Planning” section in the “System Management Overview” chapter (Chapter 1), for a sample questionnaire for determining user requirements.

The multiuser system has two interfaces, the user or interactive interface, and the programmatic interface. The user, or interactive, interface consists of the following utilities:

- GRUMP—Group and User Management Program
- KILLSES—Session Terminating Program
- SESLU—Session LU Access Table Utility

The programmatic interface consists of the set of subroutines described in the “Subroutines for Multiuser Support” chapter of the *RTE-A • RTE-6/VM Relocatable Libraries Reference Manual,* part number 92077-90037.

The Session Environment

The process of logging on, interacting with the system, and logging off, is referred to as a session. A user accesses the system by entering, at a minimum, a user name. A group name is optional. If the user wants to associate the session with a group other than the defined default group, then that group name must be specified. If the user has a password defined, it must be supplied at logon.

The system sets up an operating environment for each user session based upon the user and group accounts with which the session is associated. User and group account information is kept in user and group configuration files on the /USERS directory. Both user and group configuration files are created and modified with the GRoup and User Management Program (GRUMP).

If the user's startup program is CI, the session's associated user and group name are put in the predefined variable $LOGON in the form USER.GROUP.

After logon, the system permits only those user peripheral access requests and commands allowed within the operating environment. Users can access many peripherals with default logical unit numbers. They do not need to know system logical unit assignments. For example, each user's terminal is referred to as LU 1 rather than by the actual system LU assigned to it.

When finished with the system, the user logs off. If a user logs off via CI and a logoff program/command file has been defined, CI schedules or transfers control to that file. The system releases system resources allocated for the session and LOGON updates the user and group configuration files associated with the session. The user's total CPU usage and connect time for the session are added to the user's grand total, to the group's totals, and to the totals for the user in that group and is printed to the terminal if the user was in an interactive session.

Session Logon Process

LOGON, the session logon and logoff processor, operates in conjunction with PROMT and CM (single command version of CI). LOGON is invoked by PROMT when there is an interrupt at a terminal and no active session is operating from the terminal. CM is invoked by PROMT when there is an interrupt at a terminal and an active session is operating at the terminal. At logon, LOGON prompts the user for logon entry (USER.GROUP name) and password (if required). LOGON attempts to match the logon entry with an existing USER.GROUP definition. If it finds a match, a user ID entry is created for the session and the user's session is initiated.

LOGON uses the group configuration file to:

- Verify that the group exists.
- Verify the user is a member of the group.
- Check the group accounting limits.
- Include the group's resources in the operating environment of the user's session.

LOGON uses the user configuration file to:

- Verify the user logon name.
- Check the user's CPU usage and connect time limits within the group.
- Run the start-up program.
- Designate a working directory.
- Initialize UDSP tables.
- Create the session LU access table.
- Set the session user's capability level.
- Record the last logon time.
- Record the LU the user logged onto.
- Record the group ID logged onto by the user.
- Create the session Environment Variable Block.

LOGON creates a user ID table entry for each user session. The ID table entry contains information about the user session initially obtained from the user logon entry and later from the user and group configuration files set up in the system. The user ID table entry contains:

- User logon name.
- Session sequence number.
- Pointer to working directory.
- Terminal LU of the user or session number.
- Logoff program/command file bit.
- Number of user programs counter.
- User identification number.
- Logon time for user.
- Session CPU usage.
- Address of UDSP table in XSAM.
- ID segment address of the first session program.
- Group ID number.
- User capability level.

User-Definable Directory Search Path (UDSP)

There is a User Definable Directory Search Path (UDSP) associated with each session. A UDSP is a list specifying which directories to search when opening a file, and the order in which they are to be searched. There can be up to eight UDSPs, numbered from 0 through 8. UDSP 0 is a special case, it represents the "home" directory and has a depth of one.

The number of UDSPs and their depth (the number of entries per UDSP) are defined when the user account is created or modified. LOGON allocates space for the session UDSP in XSAM and initializes them when a user logs on. PATH is the utility that defines and displays UDSPs. The UDSP entries set by PATH are valid only for the duration of the session, they are initialized to undefined each time a user logs on.

The RU (Run Program) command in CI uses UDSP #1, and the TR (Transfer to Command File) command uses UDSP #2. Other UDSPs may be defined by the user for special use. Programmatically, D.RTR can be directed to use a UDSP when opening a file by specifying the UDSP number in the option string in an FmpOpen call.

You need to determine the optimal number and depth of paths for each user. The amount of XSAM is affected by the number of users that have UDSPs, because all UDSPs are stored in XSAM. The number of UDSPs allocated should be limited in order to conserve system available memory. Use the GRoup and User Management Program (GRUMP) to define number and depth of UDSPs. Command files can set up paths at logon.

Environment Variable Block (EVB)

The Environment Variable Block (EVB) is kept in sharable EMA and is identified by the session number; for example, Environment 90 for session 90. The block is allocated and initialized by LOGON. The size of a user's EVB is indicated in the user file. The system manager should use GRUMP to give each user an appropriate amount of space. If the user file is not modified using GRUMP, the user will not be able to export variables, aliases, or functions. However, she or he can still use them in local space.

Local CI variable space (which also contains aliases and functions) size is system configurable. This space is now in large model EMA. The default size is two pages; the effective size prior to Revision 6.0 was one page. To change this, the system manager must edit CI.LOD and change the "`em,,l`" line to "`em,n,l`", where *n* is the desired number of pages. This size will be the same for all users. See Chapter 3 for information on enlarging the EVB. Refer to the *RTE-A User's Manual*, part number 92077-90002, for descriptions of the variables.

Session LU Access Table

Each user and group definition contains a 16-word LU access table that provides a means of limiting access to LUs. If the LU number is in the LU access table, access to the LU is granted. If the LU number is not in the table, all access to the LU is denied. User and group LU access tables are created and modified with GRUMP.

Note LU 0, the bit bucket, is used for program-to-program communication and should not be removed from the LU bit map. Also make sure that all LUs corresponding to directories that users need to access (such as /PROGRAMS), and LUs corresponding to DS/NS physical links are in the LU access table.

LOGON creates the session LU access table by combining the LUs the user and associated group can access when a user logs on. The session LU access table is stored in XSAM and can be displayed and modified with SESLU (session LU access table utility). For technical detail about logon limitations when the system runs out of XSAM, see the *RTE-A System Design Manual*.

When session users try to access an LU, their LU access tables are checked. The check is done in the operating system after the LU specified in the user's I/O request has been checked against the actual system LU. If users have access to the LU, their requests are granted; otherwise, the requests are aborted. This check catches all I/O requests no matter how they are issued (with FMP calls or EXEC calls) and no matter how they are re-directed.

Note The LU access table is not part of Security/1000. Thus, user's I/O requests are checked even when Security/1000 is OFF.

Session Logoff Process

When a user has completed a session and logs off, LOGON updates the session account file with the CPU usage and connect time and de-allocates system resources for the session.

Session Utilities

To help you manage user sessions, the SESLU (Session LU) and KILLSES (Kill Session) utilities are provided. A brief description of each utility follows. Refer to Chapter 3 for a detailed description of how to use these utilities.

SESLU: Modifying and Listing Session LU Access Tables

SESLU lists and modifies session LU access tables. It does not affect the user or group configuration files associated with the session user.

A user is not allowed to remove access to a session user's terminal LU or to any LU containing the current working directory or any directories specified in the UDSP table.

If the Security/1000 system is not turned on, any user can list a session LU bit map with SESLU, but a user must be a superuser to modify a session LU bit map. If Security/1000 is turned on, the system manager can assign different required capability levels for the base function and each subfunction of the SESLU utility. Users must have the required capability to run SESLU. See Chapter 4 for a detailed description of capability levels, base functions, and subfunctions.

KILLSES: Terminating a Session

KILLSES terminates a session immediately. The user is logged off, all programs associated with the session are terminated, associated spool files are closed and released, and the user entry in the user ID table for the session is released. The session can be any type: background, interactive, or programmatic. The system session, session 0, is an exception and can never be killed.

If Security/1000 is not turned on, the user must be a superuser to run KILLSES. If Security/1000 is turned on, the system manager can assign the capability level required to run KILLSES.

Account Structure

The multiuser account system maintains two types of accounts: user accounts and group accounts. A group is a set of users who share common functions, applications, and/or resources. Group accounts are used to assign selected resources to specific sets of users and to track accounting use for the groups. User accounts provide the system with the information necessary to set up and maintain the operating environment and track accounting use for that user.

Every session user must be assigned a user account. The user account contains unique user information and information about the user for each group in which the user is a member. Unique user information pertains to the user no matter what group the user logs on with. The information about the user in each group is called USER.GROUP information and pertains to the user only when logged on associated with the group for which the information is defined. Unique user information ensures that the same set of private resources are retained in the user's operating environment regardless of associated logon group. USER.GROUP information allows tailoring of the user's environment to the application needed for the associated logon group. It also allows for finer control of accounting.

All accounts are specified to the system in the form USER.GROUP where USER and GROUP are identifiers of one to sixteen characters in length. User identifiers and group identifiers must be unique in the system.

A user can belong to several groups, but for accounting, is considered a separate entity in each. Example: Fred, logged on in group ABA, cannot switch to group CDC in mid-session. He must log off ABA before logging onto CDC.

An example account structure is shown in Figure 2-1. The sample account structure is broken into three levels: system manager, group, and user. Note that Jones is a member of three groups so that Jones can access the same private files and/or peripherals from all three groups.

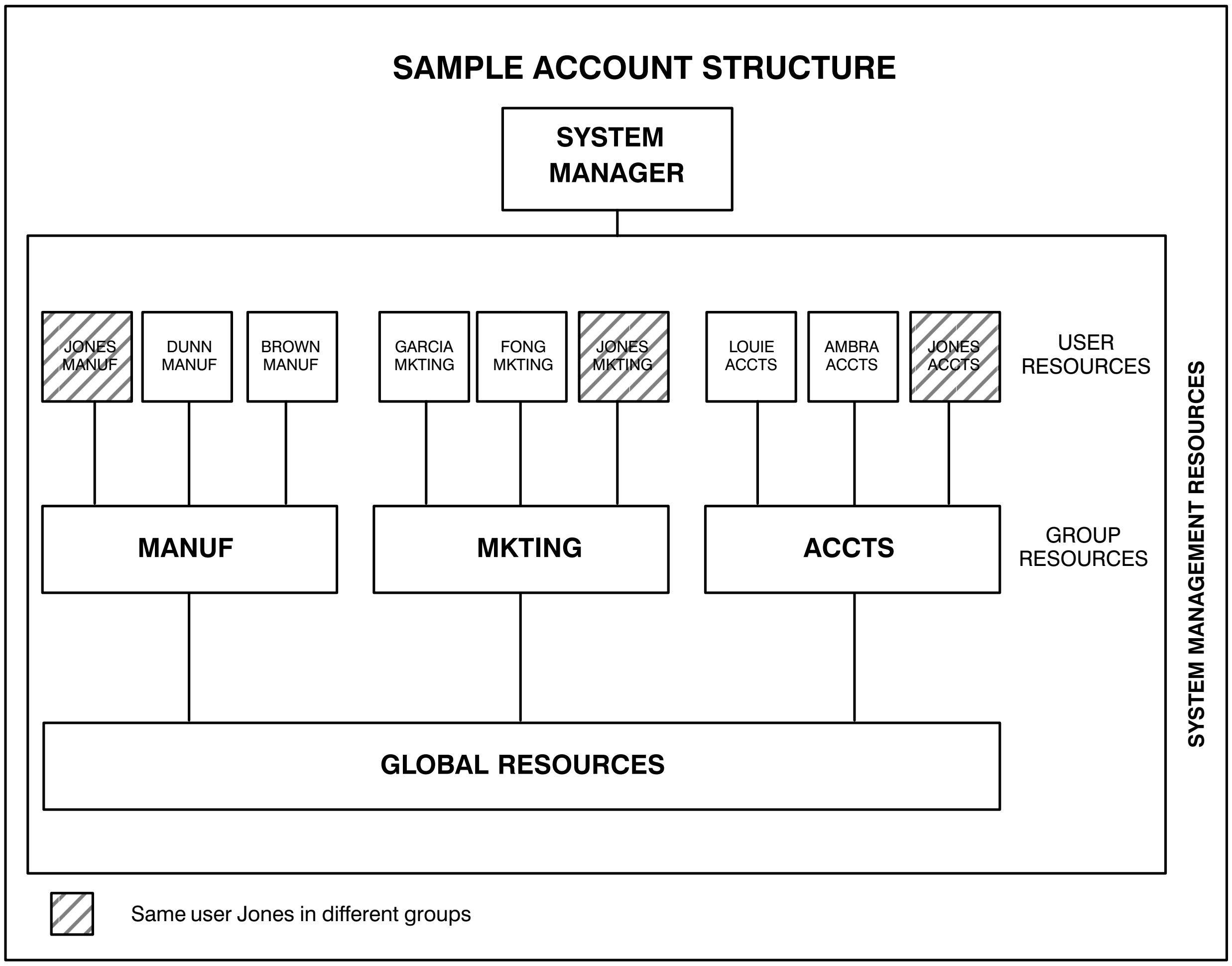

Figure 2-1. Sample Account Structure

User Account Planning

Use an account planning worksheet to list all the individual users of your system. For planning convenience, you should assign a unique identifier (up to 16 characters) to each user. A sample account matrix is shown in Figure 2-2.

GROUPS USERS	MANUF	MKTING	ACCTS
JONES	X	X	X
DUNN	X		
BROWN	X		
GARCIA		X	
FONG		X	
LOUIE			X
AMBRA			X

Figure 2-2. Sample Account Planning Matrix

Group Account Planning

After you have listed your system users, divide them into groups. Members of a group usually share one or more common attributes. Some of the criteria that may apply here are explained below.

Existing Organization

You may find it convenient to follow an existing organizational pattern. Your account structure could reflect the actual groups in your user community.

Common Files

Users who share files or data bases can be included in a group. FMGR and disk LUs can be associated with a group in such a way that they can be accessed solely by members of the group. CI volumes and directories can be assigned protection in a way that allows only group members to access them.

Common Peripherals

Groups can be formed around special peripheral access requirements. If desired, peripherals can be restricted to selected groups and users. By including the corresponding LU in the group's LU access table, peripherals may be defined to the account system so that they are automatically added to the list of peripherals individual group members may access.

Common Applications

You can separate users into groups based on their applications and job functions. Users performing similar tasks could then share related files and peripherals.

Usually, you should only form new groups when the list of users sharing a common resource is composed of users from two or more existing groups. If the users are all members of one existing group, you may be able to add the resource to that group's domain. The information gathered here will be used later on to initialize and maintain the account system.

Assign a name or an identifier, up to sixteen characters, to each group in your system. This identifier must be unique. It will be used by members of that group to identify themselves to the system. List each group in the group column of the account planning matrix, see Figure 2-2. Next, indicate the members of each group. In each group column, place an "X" in all rows corresponding to the members of that particular group. Note that there is no restriction on the number of groups to which a user may belong. This may be a requirement in situations where individuals need to access resources owned by many different groups.

/USERS Directory

Group and user accounts are managed in a multiuser account system by the Group and User Management Program (GRUMP). The multiuser account system requires a global directory, /USERS, which must contain the following:

MANAGER	User configuration file containing the user definition for the system manager who has maximum capability.
MASTERACCOUNT	System file containing logon names and corresponding user ID numbers for all users on the system.
NOGROUP.GRP	Group configuration file containing the group definition for NOGROUP. All users must belong to this group.
SYSTEM.GRP	Group configuration file containing the group definition for SYSTEM. It is used for system management purposes. MANAGER must always be a member of this group.
MASTERGROUP	System file containing logon names and corresponding group ID numbers for all groups on the system.
LOGONPROMPT	System file containing the system logon prompt.
HELP.DIR	Subdirectory that contains GRUMP help files.

These files and directories are created automatically by GRUMP if there is no /USERS directory when it is scheduled. (For more details, see "Initializing the Multiuser Account System" in this chapter). Each group and user defined in the system also has an account file residing on the /USERS directory. These files are called group and user configuration files, respectively. They are created and modified by using GRUMP.

Files on /USERS should only be modified with GRUMP or with the HP-supplied multiuser account routines and/or utilities. The files should be write protected for security reasons.

User Configuration File

The operating system maintains information on all user accounts in files called user configuration files. All user configuration files reside on the /USERS directory (which should be write protected for security), and each file name corresponds to a user's logon name. Each user configuration file contains information unique to the user regardless of associated group, and information unique to the user within each group in which that user is a member.

Caution Do not purge a user configuration file from the /USERS directory with the PU command in CI. Use the PURGE USER command in GRUMP, which purges the user from the multiuser account system with all the necessary cleanup.

The information in the configuration file is filled in during the creation of a new user account with GRUMP. The system manager, or a user with the required capability, supplies some of the information and GRUMP generates the rest. GRUMP also is used to modify or list information in the user configuration files.

Routines GetAcctInfo, ResetAcctInfo, and SetAcctLimits can be used to access and programmatically alter some of the user information. See the *RTE-A • RTE-6/VM Relocatable Libraries Reference Manual*, part number 92077-90037.

The system uses the user configuration file name to verify the user's logon name. The unique user information in the file is used by the system to:

- Verify the user's password.
- Initialize UDSP tables.
- Update the last logon time.
- Update the group ID that the user last logged on with.
- Update the LU the user last logged on to.
- Create the session LU access table.
- Determine the capability of the user.
- Determine the size of the Environment Variable Block for the user's session.
- Update the total CPU time usage of the user.
- Update the total connect time usage of the user.
- Update the last logoff time.

The information unique to the user within a group is used by the system to check that the user's CPU usage and connect time limits within the group have not been exceeded, run the startup program, designate a working directory, run the logoff program/command file, and update the CPU and connect time usage for the user within the group.

The information in the user configuration file falls into the following two categories:

1. Unique USER Information – information associated with the user regardless of group:
   - Real name.
   - Superuser bit.
   - Encoded password.
   - User ID number.
   - UDSP number and depth.
   - Block number of the first group record (internal information).
   - Number of entries in the user's group list (internal information).
   - Default logon group.
   - Last logoff time.
   - Last logon time.
   - Group ID number with which the user last logged on.
   - LU the user logged on to last.
   - LU access table.
   - Size of the Environment Variable Block.
   - Capability level.
   - Total CPU usage for user in all groups.
   - Total connect time for user in all groups.

2. USER.GROUP Information – information associated with the user only when logged on with the group for which it is defined:
   - Startup program to run.
   - Logoff program/command file.
   - Working directory name.
   - CPU usage and connect time totals within a specific group.
   - CPU usage and connect time limits within a specific group.

The information is divided this way:

- For a user as a single entity—one real name, one password, one user ID number, and comprehensive CPU usage and connect time totals.

- To enforce the definition that one user has one set of resources—one LU access table and one UDSP number and depth.

- To enforce security—one capability level.

- To enable the system manager to tailor the working environment of the user to the application needed for that user in each group—one startup program, one logoff program/command file, and one working directory per group.

- For finer control of accounting and accounting limits—CPU usage and connect time totals, and limits for the user in each group.

MASTERACCOUNT File

The MASTERACCOUNT file is a protected system file on the /USERS directory. This file is created by GRUMP during the initialization phase and contains the logon name and corresponding user ID number for all system users. It should be write protected for security.

The first record contains system information with each of the remaining records containing the logon name of a system user. The contents of the record is the user logon name (maximum of 16 characters). The user identification number is the same as the record number (example, user ID 26 is record number 26 in the MASTERACCOUNT file) so it is easy to determine the user name given the user ID. If you have the user ID number and need to know the name of any user, use the subroutine IDToOwner.

Given the user name, this file is not searched to determine the user ID because users may have been deleted or there may have been an unrecoverable error in the creation of a new user. In this case, use subroutine OwnerToID.

Group Configuration File

The operating system maintains information on all group accounts in files called group configuration files. The file resides on the /USERS directory (which should be write protected for security) and its name corresponds to the group's logon name with the type extension .GRP. This distinguishes it from user configuration files on /USERS.

Caution Do not purge a group configuration file from the /USERS directory with the PU command in CI. Use the PURGE GROUP command in GRUMP, which purges the group from the multiuser account system with all the necessary cleanup.

The information in the group configuration file is filled in during the creation of a new group account with GRUMP. The system manager, or a user with the required capability, supplies some of the information and GRUMP generates the rest. For example, GRUMP uses the MASTERGROUP file to assign the group identification number when the new group is created. GRUMP also can be used to list or modify information in an existing group configuration file.

Routines GetAcctInfo, ResetAcctTotals, and SetAcctLimits can be used to programmatically access and alter group information. See the *RTE-A • RTE-6/VM Relocatable Libraries Reference Manual* for information about these routines.

Each group configuration file contains the following information:

- Group identification number.
- Group totals for CPU usage and connect time.
- Group limits for CPU usage and connect time.
- Group LU access table.
- Number of records in its members list (internal information).
- List of member records.

The system uses this file when it creates a user session to check that the user is a member of the group, that the group accounting limits have not been exceeded, and to include the group's resources in the operating environment of the user's session. When a group member logs off, the CPU usage and connect time totals are updated in the group configuration file.

MASTERGROUP File

The MASTERGROUP file is a protected file on the /USERS directory. This file is created by GRUMP during the initialization phase and contains the logon name and corresponding group ID of all the groups on the system.

The first record contains the last group identification number assigned by the system and other system information. The remaining records contain the logon name of each group defined in the system. A group's identification number is its corresponding record number in the MASTERGROUP file, with the maximum group ID number being 2047. The contents of the record is the group's logon name (maximum of 16 characters), so it is easy to determine the group name given the group ID. If there is a need to know the name of a group for which the group ID is known, use subroutine IdToGroup.

This file is never searched to determine the group ID given the group name. If there is a need to know the ID of a group for which the group name is known, use subroutine GroupToID.

NOGROUP and Default Logon Group

One requirement of the multiuser account system is that all users belong to group NOGROUP. NOGROUP is a group from which users gain no resources. Its purpose is to allow systems to operate without setting up groups. To operate without assigning groups, all that needs to be done is to make NOGROUP the default logon group for all user accounts by using GRUMP.

If the system has been set up to assign users to groups, and you do not want a particular user logging on with a group assignment of NOGROUP, you can set that user's connect time limit within NOGROUP to zero. If you do not want any users to log on with a group assignment of NOGROUP, set the connect time limit for NOGROUP to zero.

All user account definitions contain a list of groups (one or more) to which the user belongs. Any one of the groups may be declared the default logon group by using GRUMP. This allows users to log on without specifying a group account name in the logon entry while automatically associating the session with the default logon group.

Initializing the Multiuser Account System

The Group and User Management Program (GRUMP) creates and manages the multiuser account system. The account system requires a directory called /USERS, which contains a subdirectory HELP and the files MASTERACCOUNT, MASTERGROUP, LOGONPROMPT, MANAGER, NOGROUP.GRP and SYSTEM.GRP. All of these are created automatically by GRUMP if there is no /USERS directory when it is scheduled. GRUMP initializes the multiuser account system in these steps:

1. Prompts for the LU where the directory /USERS is to reside.
2. Creates directories /USERS and /USERS/HELP (for the GRUMP help files).
3. Creates the MASTERACCOUNT file.
4. Reserves record number two and three in the MASTERACCOUNT file for SYSTEM and MANAGER, respectively.
5. Sets the last assigned user identification number to 3 in the MASTERACCOUNT file.
6. Initializes the LOGONPROMPT file. The default is:

   Please log on:
7. Creates the MASTERGROUP file and sets the last assigned group identification number to 2. Records 3 through 2048 are initialized to zeros.
8. Creates the group configuration file for NOGROUP and sets all the attributes that can never be modified; ID #0, no LUs set in the LU access table, and MANAGER is a member. The CPU and connect time limits are set to infinity but can be modified.
9. Creates the group configuration file for SYSTEM and sets all the attributes that can never be modified; ID #2, no CPU or connect time limits, and MANAGER is a member. All LUs are put in the LU access table, but they can be modified.
10. Creates a user configuration file for MANAGER and sets the following attributes that can never be modified; capability level of 31, all LUs put in LU access table, create group entries for NOGROUP and SYSTEM. It then prompts to see if the operator wants to make any modifications to the MANAGER account definition.

Re-Initializing the Multiuser Account System

Caution Read all instructions carefully. Unless done carefully and in correct sequence, re-initializing the multiuser account system can prevent you from logging on to your system.

Do not log off after you have purged /USERS and before you have re-initialized the multiuser account system or you will not be able to log on again.

If you already have a multiuser account system set up, but want to re-initialize from scratch, follow these steps:

1. Make sure no other users are logged on.
2. Copy all the GRUMP help files from /USERS/HELP to a temporary directory with the purge option.
3. Purge everything in /USERS.
4. Purge the /USERS directory.
5. Run GRUMP immediately to re-initialize the multiuser account system.
6. Copy the help files from the temporary directory to /USERS/HELP that GRUMP created during the initialization.
7. Create the new multiuser system.

Note You will have to redefine ownership and associated groups for almost all volumes and directories because they are based on user and group identification numbers that very likely have changed. Groups NOGROUP and SYSTEM are always guaranteed the same group identification numbers. User SYSTEM will always have the same user ID. User MANAGER will always have user ID 3 when created with GRUMP, but could have a different user ID if it was created prior to Revision 5000.

3

GRUMP, SESLU, and KILLSES Utilities

GRUMP Utility

The GRoup and User Management Program (GRUMP) is a command-driven utility for managing a multiuser account system. It is part of the VC+ multiuser product.

Commands are run from a command file or interactively from a terminal. The operation mode of GRUMP can alternate between interactive and non-interactive with the TR(ansfer) command. The TR command can be issued any time GRUMP is waiting for input.

GRUMP produces two kinds of messages: error messages printed to the terminal (and a log file if one is specified) to indicate that an error has occurred, and information messages printed to a specified log file (and the terminal if the quiet option is not on) to describe what is happening when running GRUMP.

The RINFO (Reset Multiuser Accounting Information) and SINFO (Show Multiuser Accounting Information) utilities that reset and display CPU usage and connect time should be used only on systems that are not using groups. If you use RINFO in a system with groups, it resets the information for the unique user and USER.NOGROUP, which may not be what you want. Use GRUMP to reset and display CPU usage and connect time for systems using groups. The RINFO and SINFO utilities are described in Appendix I.

Caution Do not remove GRUMP's ID segment (OF GRUMP) because this can corrupt multiuser files.

Note GRUMP's scheduled name may change temporarily to GRMP0, GRMP1,...or GRMP9 during execution. GRUMP tries to maintain the scheduled name as much as possible.

Running GRUMP

The GRUMP runstring is as follows:

```
   [RU] GRUMP [InputSource [LogFile]] [+Q]
```

where:

InputSource	user's terminal LU or file descriptor for a command file. If not specified or if a user's terminal LU is specified, GRUMP enters interactive mode. If a file descriptor is specified, GRUMP executes the command file.
LogFile	user's terminal LU or file descriptor to which all prompts, responses, error and information messages are to be written. If not specified, all prompts, responses, and messages are printed to your terminal LU.
`+Q`	quiet option that suppresses information messages.

The parameters are position dependent; therefore, an unspecified parameter must be delimited with commas. For example, to default the *InputSource* parameter to your terminal LU and specify MYFILE as the log file, you would enter the following command:

```
   CI> grump,myfile
```

The +Q option must be the last parameter in the runstring if an input source and log file are specified. Space holders for input source and log file need not be entered if +Q is the only option used.

Except for responses, everything written to the log file is preceded by an asterisk-blank (*) to make the line a comment line. Comment lines are not executed if the log file is later used as an input source. TR commands are preceded by asterisk-arrows (* >>) so they become comments indicating the source of subsequent commands.

Interactively

To run GRUMP interactively, enter the GRUMP runstring without specifying an input source; for example:

```
   CI> grump
   GRUMP>
```

In interactive mode, you can enter a GRUMP command after each GRUMP prompt. If the GRUMP command you enter has required parameters, you can specify values for the parameters when entering the command. If you do not specify any or all parameter values, GRUMP prompts you to enter the unspecified values.

GRUMP uses the standard RTE command stack. This supports various operations such as finds, page movement, line marking, and so on. Refer to the *RTE-A User's Manual* for information on command stack usage.

The following example shows the start of creating a new group called MAX:

```
CI> grump
GRUMP> ne g
Enter Group Logon Name: max
.
.
.
```

In the following example, a new user is created with the name “Jamala”, real name “James Alabama”, password “pup”, capability level 12, and UDSP 4:4.

```
CI> grump
GRUMP> ne u jamala ‘James Alabama‘ pup 12 /e 4:4

Creating user JAMALA


All users must be in group NOGROUP.


Enter information for JAMALA.NOGROUP


Enter working directory name [::JAMALA]:
```

If you are entering commands interactively and GRUMP encounters an error (an illegal value in a CPU usage limit, for example) GRUMP prints an error message and again prompts you for the value or information.

Using a Command File

To run GRUMP with a command file, enter the GRUMP runstring and specify a file containing GRUMP commands as the input source; for example:

```
CI> grump newacct
```

Non-interactive commands are supplied by command files and no prompts are issued. Use caution when working with command files because errors within them can cause unexpected results. You should always use a log file in conjunction with your command file to help you check results and trace errors.

If you are using a command file, GRUMP prints errors to the screen and log file and tries to continue. If GRUMP is unable to recover from the error, it reports the error, flushes the current input string, goes interactive, and again prompts for the input. Use the log file to locate the errors in the command file. After correcting the command file, use GRUMP to undo any changes made by the command file before reexecuting the command file.

Appendix C contains an example of a log file used in conjunction with a command file.

GRUMP Command Summary

GRUMP commands fall into five categories:

- General Commands.
- Commands for Adding Accounts.
- Commands for Modifying Accounts.
- Commands for Displaying Account Information.
- Commands for Purging Accounts.

Tables 3-1 through 3-5 group GRUMP commands by function.

Table 3-1. General GRUMP Commands

Command	Purpose	Page
/A	Aborts current command	3-7
EX	Terminates GRUMP	3-15
HE or ?	Lists valid commands and help for individual commands	3-16
KI	Immediately terminates current user session	3-16
RU	Clones and runs a program from GRUMP	3-30
TR	Transfers control from one LU or file to another	3-30

Table 3-2. GRUMP Commands for Adding Accounts

Command	Purpose	Page
NE G	Creates a group configuration file for a new group	3-21
NE U	Creates a user configuration file for a new user	3-23

Table 3-3. GRUMP Commands for Modifying Accounts

Command	Purpose	Page
AL G	Alters attributes defined for groups	3-7
AL U	Alters attributes defined for users	3-9
PA	Alters user password in the user configuration file	3-27
RE G	Resets group accounting information	3-29
RE U	Resets user accounting information	3-29

Table 3-4. GRUMP Commands for Displaying Account Information

Command	Purpose	Page
LI G	Lists one or more group account entries	3-16
LI U	Lists one or more user account entries	3-17

Table 3-5. GRUMP Commands for Purging Accounts

Command	Purpose	Page
PU G	Removes a group from the accounting system	3-27
PU U	Removes a user from the accounting system	3-28

Command Conventions

The following conventions apply to all GRUMP commands:

- Only the first, or first and second, characters of the command are checked; all other characters are ignored.
- GRUMP prompts for required command characters and parameters that are not entered at the GRUMP> prompt. For example,

```
GRUMP> ne
Enter (G)roup or (U)sers:

GRUMP> ne g
Enter group logon name:

GRUMP> new g general
Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit]:
```

- All GRUMP commands must be entered at the GRUMP prompt. /A, /HE, and /TR can also be entered at the parameter prompts.
- Multiple GRUMP commands cannot be entered in a single input string.
- One command and all its parameters can be entered in a single input string at the GRUMP prompt. The parameters must be entered in the proper order. If an error is encountered, the rest of the string is ignored, and GRUMP tries to recover. GRUMP prompts again for any omitted values.
- Enclose character strings with embedded commas or blanks in back quotes (‘ ‘), or GRUMP reads the blanks and commas as parameter separators. Back quotes can be omitted if the string is the only input on a line or is the last parameter in a multiple value string. Use caution when omitting back quotes. The following examples all give the same result (user input is underlined):

```
1. GRUMP> ne u jamala ‘James Alabama‘ pup 12 /e 4:4 2

2. GRUMP> ne u jamala
   Enter user’s real name : James Alabama
   Enter user’s password: pup 12 /e
   Enter #UDSPs:depth : 4:4
   Enter the size of the Environment Variable Block in pages : 2

3. GRUMP> new u jamala James Alabama
   Enter user’s password: pup 12 /e 4:4 2
```

The result of each example is the creation of user JAMALA with:

real name	James Alabama
password	pup
capability level	12
LU access table	all bits set
UDSPs:depth	4:4
EVB size in pages	2

GRUMP Commands

Abort (/A)

Purpose: Aborts the current command or subfunction within the ALTER USER command.

Syntax: `/A` or `CTRL-D`

Description:

/A is identical to EXIT when entered at the GRUMP> prompt (global command).

Note that you must use CTRL-D to abort from the prompt “Enter working directory name[xx]:” because /A is considered a valid response. In other words, /A is interpreted as a working directory name.

If /A is entered within any command except ALTER USER, the command has no effect on the related multiuser files. See the ALTER USER command for its different use of the /A command.

Alter Group (AL G)

Purpose: Modify the attributes defined for a group.

Syntax: `AL G` *group*

group Name of the group account to be modified. Legal values for group are:

groupName	modify the group account specified.
`@`	modify all existing groups.

Description:

The modifications you make with ALTER GROUP do not affect users currently logged on, only users logging on associated with that group after the alterations have been made to the group configuration file.

GRUMP prompts for values for all the attributes that can be modified with this command. Current values are listed as defaults with [X...X] meaning default.

- Group name.
- CPU usage.
- Connect time limits.
- LU access table.

GRUMP returns the message “Capability level not high enough to alter group...” when you do not have the required capability to change an attribute and proceeds to the next attribute.

The prompts for ALTER GROUP are:

```
Enter group logon name:
Enter new group logon name [XXXXX]:
Enter CPU limit (hh:mm:ss or –1 for No Limit) [XX]:
Enter connect time limit (hh:mm:ss or –1 for No Limit) [XX]:
LU access table modifications ( [–]LU#[:LU#2] ):
```

Group Name

This is the group logon name. The name must be unique, follow file naming conventions, and be one to sixteen characters with no parentheses. This definition is not case sensitive so either upper or lowercase characters can be used.

CPU Usage Limit

This defines the CPU limit for the group. The values for CPU limit are:

hh:*mm*:*ss*	*hh* − hours (up to 5960).
or	*mm* − minutes (up to 59).
hh:*mm*	*ss* − seconds (up to 59).
0	inhibits CPU use by group.
−1	group has unlimited use of CPU.
<cr>	no change to value.

No more members may log on after the CPU limit has been exceeded until you rectify the situation by recording the usage and resetting the usage with the RESET GROUP command or by altering the CPU limit for the group. Users logged on when the limit is exceeded are not affected.

Connect Time Limit

This defines the connect time limits for the group. The values for connect time limits are:

hh:*mm*:*ss*	*hh* − hours (up to 596500).
or	*mm* − minutes (up to 59).
hh:*mm*	*ss* − seconds (up to 59).
0	inhibits group from logging on.
−1	gives group unlimited connect time.
<cr>	no change to value.

No more members may log on if the connect time limit has been exceeded until you rectify the situation by recording the connect time and resetting it with the RESET GROUP command or by altering the connect time limit for the group. Users logged on when the connect time limit is exceeded are not affected.

LU Access Table

The LU access table defines the LUs to which the group has access. The values are:

n	adds LU *n* to LU access table.
−*n*	removes LU *n* from LU access table.
m:*n*	adds LU *m* to LU *n* to LU access table.
−*m*:*n*	removes LU *m* to LU *n* from LU access table.
`/L`	lists LUs set in LU access table that group can access.
`/E`	ends LU access table modifications.
<cr>	ends LU access table modifications if sole input at prompt.

GRUMP continues to prompt for alterations until a /E is encountered in an input string or a <cr> is entered at the LU access table modifications prompt.

Multiple input must be separated by a comma or space as in this example where access to LUs 5, 6, 10 through 20, and 30 is given, and access to LUs 40 and 50 through 55 is removed:

```
LU access table modifications ( [ – ]LU#[:LU#2] ): 5 6 10:20 30 –40 –50:55
```

Alter User (AL U)

Purpose: Modify certain attributes of an existing user account.

Syntax: `AL U` *UserGroup*

UserGroup User and group information. Can be specified as follows:

user	Information unique to user and certain attributes in the user.NOGROUP record.
user.	Information unique to user regardless of group. It is used to add a user to a group and change the default logon group.
user.group	User.group information about user within specified group.
@.group	Corresponding user.group information of all members of the specified group. GRUMP loops through the list of members and modifies member attributes one at a time.
user.@	Group attributes of user in all groups in which the user is a member. GRUMP loops through user's group list and modifies the attributes in each of the user.group records one at a time.

Description:

You can use GRUMP to modify the attributes for which you have the required capability. If the user does not have the required capability level, GRUMP returns the message "Capability level not high enough to alter user..." and proceeds to the next attribute.

If an attribute (such as the name of the SYSTEM group) can never be modified, GRUMP returns the message "XXXXX attribute can never be modified" and proceeds to the next attribute.

Invalid input is ignored, and the prompt is re-issued. If invalid input is read from a transfer file, GRUMP goes into the interactive mode before re-issuing the prompt.

To add a user to a group or change the default logon group, enter "USER." as the USER.GROUP parameter. GRUMP prompts for modifications to the unique user attributes. Then it prompts for the group name(s) to which the user should be added, the USER.GROUP information needed for each group, and the default logon group.

You cannot remove a user from a group with the ALTER USER command. Use the PURGE USER command to do this.

Use of /A in the ALTER USER command differs from use of /A in the other commands because ALTER USER is divided into four steps:

1. Altering unique user information.
2. Altering information for a user within a group in which the user is a member.
3. Adding a user to a group where not a member and defining the user attributes within that group.
4. Changing the default logon group.

If you have completed one or more steps and enter a /A within a subsequent step, modifications made in the step you are in are not acted upon and the ALTER USER command is aborted. Modifications made in any previous step are permanent. Also, you may add the user to several groups in Step 3. Once you have defined all the information for the user within a group, the user cannot be removed from that group by a subsequent /A in Step 3 or Step 4.

You cannot modify the capability level, CPU and connect time limits, and LU access table for the user MANAGER.

You can use the ALTER USER command to:

- Modify unique user information
  - Logon name
  - Real name
  - Password
  - Capability level
  - LU access table
  - Number and depth of UDSPs
  - Size of the EVB
- Modify user information within a group
  - Default working directory
  - Startup command
  - Logoff program/command file
  - CPU and connect time limits
- Add a user to a group
- Define default logon group

See the examples at the end of the "Alter User" section for the command prompts.

Unique User Information

Logon Name

This is the logon name for the user. The logon name must be unique, follow the file naming conventions, and be one to sixteen characters with no parentheses. This definition is not case-sensitive, therefore, either uppercase or lowercase characters can be used.

Real Name

This is the user's real name. The real name may not be more than thirty characters. This definition is case sensitive. The name must be enclosed in back quotes (' ') if it is a parameter in a multiple parameter input string.

Password

You can define a password for the user. The password can be no more than fourteen characters with no commas or blanks. Special characters are not recommended. GRUMP issues the following prompt if a password is already defined and you enter a <cr> (indicating no password) at the enter password prompt:

```
   Change password to no password (Yes/No) [N]:
```

This prevents accidentally erasing an existing password. You are not required to know an existing password to change it.

Capability Level

This is the user's capability level (USERCPLV). The capability level is an integer value from 0 (lowest) through 31 (highest). Capability level 31 sets the superuser bit in the user configuration file. A capability level of 0 to 30 clears the superuser bit in the user configuration file. The default capability level is 10. You can only assign capability levels that are equal to or lower than your own.

LU Access Table

The LU access table defines the LUs that can be accessed. You can remove or add any LUs from the user's LU access table. The values are:

n	adds LU *n* to LU access table.
−*n*	removes LU *n* from LU access table.
m:*n*	adds LU *m* to LU n to LU access table.
−*m*:*n*	removes LU *m* to LU *n* from LU access table.
`/L`	lists LUs set in LU access table that user can access.
`/E`	ends LU access table modification.
<cr>	ends LU access table modifications if sole input at prompt.

GRUMP continues to prompt for alterations until a /E is encountered in an input string or a <cr> is entered at the LU access table modifications prompt.

Multiple input must be separated by a comma or space as in this example where access to LUs 5, 6, 10 through 20, and 30 is given and access to LUs 40, and 50 through 55 is removed.

```
LU access table modifications ( [ – ]LU#[:LU#2] ): 5 6 10:20 30 -40 -50:55
```

Be sure the user has access to LU 0, used for program-to-program communmication. Also, make sure the user can access the LUs containing directories needed such as /PROGRAMS and the default working directory.

Number and Depth of UDSPs

This defines the number and depth of the User-definable Directory Search Paths for this user. A <cr> leaves the values as they are. The number and depth of UDSPs must be separated by a colon with no spaces between the numbers and the colon. The legal values are:

m:*n*	both zero or both not zero (*m* = number, *n* = depth).
m:	modify number only.
:*n*	modify depth only.
<cr>	use the default (value is unchanged).

Environment Variable Block

This is the size of the Environment Variable Block (EVB) in pages, which is allocated to the user's session at logon. The size can be set to any value from 0 pages to 32 pages, inclusive.

GRUMP prints a message indicating that unique user information was modified and terminates unique user information modification. Any changes made up to this point are permanent.

Default Working Directory

This defines the default working directory for the user within the group. A <cr> leaves the value as it is. The directory name must not exceed 63 characters including delimiters and must follow the file naming conventions. GRUMP does not verify that the named directory belongs to the user whose account is being modified. The LU containing the working directory must be in the user's LU access table.

If the directory specified does not exist, a GRUMP prompt asks whether the directory should be created.

```
   Create directory XXXXX (Yes/No) [N]:
```

If the directory is to be created, GRUMP prompts for the LU on which it should be created.

```
   What LU should the directory go on [0]:
```

The default LU, 0, is the same as the current working directory or the lowest numbered disk LU on which directories can be created.

Startup Command

This defines the startup command (for the user in the group), which is the program to be executed when the user logs on. The LU containing the directory on which the logoff program/command file must be in the user's LU access table. A <cr> leaves the value as it is. The command can have a maximum of 80 characters.

```
Enter the startup command [RU CI.RUN::PROGRAMS]: ru ci.run::programs logon.cmd
```

If the program to be run at logon is RP'ed with the D option in the Welcome file to increase system performance, it must be specified without a path name at the startup command prompt to force the system to use the Proto-ID in XSAM instead of creating one from scratch. Thus, in the above example, the command

```
   ru ci.run logon.cmd
```

should be specified, not "`ru ci.run::programs logon.cmd`".

Starting programs other than CI must be loaded and in the directory specified in the command. A startup command string must be enclosed in back quotes (‘) if it is a parameter in a multiple parameter string. For example:

```
GRUMP> alter user jamala.lab , ‘ru ci.run::programs logon.cmd‘ , , 100:0:0
```

Logoff Program/Command File

This defines the logoff program/command file (for the user in the group) to be scheduled or to which execution is transferred when the user exits CI. The logoff program must be loaded and on the directory specified. The logoff command file's directory location must be specified. The LU having the directory where the logoff program/command file is located must be in the user's LU access table. If Security/1000 is OFF, the protection of /USERS and the corresponding user configuration file must be RW/R (if groups are not being used) or RW/R/R (if groups are being used) in order for this feature to work.

The logoff program/command file must be enclosed in back quotes (‘ ‘) if it is a parameter in a multiple parameter string. A <cr> at the prompt specifies that the logoff program/command file is undefined. If a logoff program/command file is defined and you enter a <cr> at the ‘Enter Logoff program/command file’ prompt, GRUMP prompts to ask if the logoff program/command file should be changed to undefined.

```
Enter the logoff program/command file [RU,CLEANUP.RUN::FRED]: <cr>
Change the logoff program/command file to UNDEFINED (Yes/No) [N]:
```

CPU Usage Limit

This defines the CPU limit for the user within the group. The values for the CPU limit are:

hh:*mm*:*ss*	*hh* − hours (up to 5960).
or	*mm* − minutes (up to 59).
hh:*mm*	*ss* − seconds (up to 59).
0	inhibits CPU use by group.
−1	group has unlimited use of CPU.
<cr>	no change to value.

If the USER.GROUP CPU usage limit has been exceeded, the user cannot log on associated with the group until the situation has been rectified.

Connect Time Limit

This defines the connect time limits for the user within the group. The values for connect time limits are:

hh:*mm*:*ss*	*hh* − hours (up to 596500).
or	*mm* − minutes (up to 59).
hh:*mm*	*ss* − seconds (up to 59).
0	inhibits group from logging on.
−1	gives group unlimited connect time.
<cr>	no change to value.

If the USER.GROUP connect time limit has been exceeded, the user cannot log on associated with the group until the situation has been rectified.

Adding a User to a Group

If USER. was entered, GRUMP prompts for unique user information and then prompts if you want to add the user to any existing groups. If you enter YES, GRUMP prompts for a group name and USER.GROUP information for the user in the specified group. A /E or <cr> at the “Enter group name” prompt terminates prompts for groups. A /A aborts the rest of the ALTER USER command but previous modifications are permanent.

Default Logon Group

This is the name of the user's default logon group. If USER. was entered, GRUMP prompts for the unique user information, existing groups to which the user should be added and then for the default logon group for the user. The user must already be a member of the group that is specified. To change the default logon group, the user name must be followed by a period. For example, JAMALA. is a correct user name. If the period is omitted, the default logon group question is not asked. A /A aborts the rest of the ALTER USER command but previous modifications are permanent.

Alter User Command Examples

If the USER.GROUP parameter is not supplied, the first prompt for all cases is "Enter user.group parameter:".

1. Altering "USER"

```
 GRUMP> al us jamala
 Enter new user logon name [JAMALA]:
 Enter user's real name [James Alabama]:
 Enter password (a <CR> gives no password):
*Change password to no password (Yes/No) [N]
 Enter capability level (31=SU) [10]:
 LU access table modifications ( [–]LU#[:LU#2] ):
 Enter #UDSPs:depth [4:4]:
 Enter the size of the Environment Variable Block in pages [2]:
 Enter working directory name [::JAMALA]:
*Create directory ::JAMALA (Yes/No) [N]:
*What LU should the directory go on [0]:
 Enter the startup command [RU CI.RUN::PROGRAMS]:
 Enter the logoff program/command file [LOGOFF.CMD::JAMALA]:
*Change the logoff program/command file to UNDEFINED (Yes/No) [N]:
 Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit]:
 Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit]:
```

* This prompt is dependent upon the response to the prompt immediately preceding it. See the discussion of individual terms for information on the occurrence of asterisked prompts.

2. Altering “USER.” (Note the trailing period)

```
 GRUMP> al us jamala.
 Enter new user logon name [JAMALA]:
 Enter user’s real name [James Alabama]:
 Enter password (a <CR> gives no password):
*Change password to no password (Yes/No) [N]:
 Enter capability level (31=SU) [10]:
 LU access table modifications ( [–]LU#[:LU#2] ):
 Enter #UDSPs:depth [4:4]:
 Enter the size of the Environment Variable Block in pages [2]:

 Do you wish to include the user in any existing
 group other than NOGROUP (Yes/No) [N]:
*Enter group name (/E or <CR> to end):

 Which group should be the default logon group [NOGROUP]:
```

* This prompt is dependent upon the response to the prompt immediately preceding it. See the discussion of individual terms for information on the occurrence of asterisked prompts.

3. Altering “USER.GROUP”

```
 GRUMP> al us jamala.nogroup
 Enter working directory name [::JAMALA]:
*Create directory ::JAMALA (Yes/No) [N]:
*What LU should the directory go on [0]:
 Enter the startup command [RU CI.RUN::PROGRAMS]:
 Enter the logoff program/command file [LOGOFF.CMD::JAMALA]:
*Change the logoff program/command file to UNDEFINED (Yes/No) [N]:
 Enter CPU limit (hh:mm:ss or –1 for No Limit) [1:00:00]:
 Enter connect time limit (hh:mm:ss or –1 for No Limit) [500:00:00]:
```

* This prompt is dependent upon the response to the prompt immediately preceding it. See the discussion of individual terms for information on the occurrence of asterisked prompts.

Exit (EX)

Purpose:	Terminates GRUMP.
Syntax:	`EX` `/E`

Help (HE or ?)

Purpose:	Displays a summary of GRUMP commands or a brief description of a specific command.
Syntax:	`?[?] [`*command*`]`
	`[/]HE [`*command*`]`

command	Specific GRUMP command to be explained, default is a list of possible GRUMP commands.

Description:

HELP, /HELP, ??, and ? are identical except that /HELP, ??, and ? may be entered from within GRUMP commands while HELP may be entered only at the GRUMP> prompt.

GRUMP does not prompt for the optional command parameter. It lists all the commands. Additional input on the line is read as a command parameter. For example,

```
Enter password : /HE /TR cmdfile
```

results in a description of the TRANSFER command and sets the password to cmdfile. It does not list all commands and then transfer to cmdfile.

To get help from the "Enter working directory name" prompt, you must use ? rather than /HE because /HE is considered a valid response. In other words, "/HE" is interpreted as a working directory name.

Kill Session (KI)

Purpose:	Terminates a session.
Syntax:	`KI` *session* `[OK]`

session	Session identifier of user to be logged off.
`OK`	Optional parameter to override the verification prompt.

Description:

The KI command logs off a particular user session, removes all programs associated with the session, closes and releases the associated spool files, and releases the user ID entry in the User Table for the session. Session 0, the system session, cannot be killed. KI cannot be used to kill a session that is also running GRUMP. You must be a superuser to use this command.

List Group (LI G)

Purpose:	Lists the account information in the configuration file for the group(s) specified
Syntax:	`LI G` *group* `[`*ListFile*`]`

group	Group name whose configuration file information is to be listed, @ may be specified to indicate all group accounts.
ListFile	Name of a file or terminal LU to which the listing should be sent, default is the terminal.

Example of the List Group Command

```
GRUMP> li g accounting

*********************************************************************

Group:                   ACCOUNTING
Group ID:                26
Total CPU Time:              1 HR  57 MIN  24 SEC 740 MSEC
Total Connect Time:         44 HR   1 MIN  50 SEC   0 MSEC
CPU Limit:               No Limit
Connect Time Limit:      No Limit

LU Access Table:
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255

Members:
   SANDI          SAM          JAMALA          MAX
   HENRY          JUDY

*********************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table.

List User (LI U)

Purpose: Lists the account information in the configuration file for the specified users.

Syntax: `LI U` *UserGroup* `[`*ListFile*`]`

UserGroup	User and group information. Can be specified as follows:
user	Unique user information and certain fields in the user.NOGROUP record (for systems not using groups).
user.	Unique user information (regardless of group).
user.group	Unique user information and user.group information about user within the specified group.
@.group	Corresponding user.group information for all members of the specified group. GRUMP lists the group information, then loops through the list of members listing the user.group information for the members one at a time.
user.@	User.group information for the user in all groups in which the user is a member. GRUMP lists the unique user information, then loops through the user's group list listing each of the user.group records one at a time.
ListFile	Name of a file or terminal LU to which the listing should be sent, default is the terminal.

Examples of the List User Command

1. Example of “USER”

```
GRUMP> li u jamala

*****************************************************************

User:                          JAMALA
Real Name:                     James Alabama
User ID:                       527
UDSPs:Depth:                   4:4
EVB size in pages:             2
Capability Level:              10
Default Logon Group:           NOGROUP
Startup Program:               RU CI.RUN::PROGRAMS
Working Directory:             ::JAMALA
Logoff Program/Cmndfile:       [LOGOFF.CMD::JAMALA]:
Last Logon Time:               Thu Jan 2, 1990 10:00:00 am
Last Logoff Time:              Thu Jan 2, 1990 12:05:31 pm
Group ID Last Logged on:       0
LU Last Logged onto:           101
Total CPU Time:                   1 HR  57 MIN  24 SEC  740 MSEC
Total Connect Time:              44 HR   1 MIN  50 SEC    0 MSEC
CPU Limit:                     No Limit
Connect Time Limit:            No Limit
LU Access Table:
     0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255

*****************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table

2. Example of “USER.”

```
GRUMP> li u jamala.

*****************************************************************

User:                          JAMALA
Real Name:                     James Alabama
User ID:                       527
UDSPs:Depth:                   4:4
EVB size in pages:             2
Capability Level:              10
Default Logon Group:           NOGROUP
Last Logon Time:               Thu Jan 2, 1990 10:00:00 am
Last Logoff Time:              Thu Jan 2, 1990 12:05:31 pm
Total CPU Time:                    1 HR  57 MIN  24 SEC  740 MSEC
Total Connect Time:               44 HR   1 MIN  50 SEC    0 MSEC
LU Access Table:
     0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255

Groups:
   NOGROUP        MANUALS           ACCOUNTING

*****************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table.

3. Example of “USER.GROUP”

```
GRUMP> li u jamala.nogroup

*****************************************************************

User:                        JAMALA
Real Name:                   James Alabama
User ID:                     527
UDSPs:Depth:                 4:4
EVB size in pages:           2
Capability Level:            10
Default Logon Group:         NOGROUP
Last Logon Time:             Thu Jan 2, 1970 10:00:00 am
Last Logoff Time:            Thu Jan 2, 1970 12:05:31 pm
Total CPU Time:                   1 HR  57 MIN  24 SEC  740 MSEC
Total Connect Time:              44 HR   1 MIN  50 SEC    0 MSEC
LU Access Table:
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  **  15
     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
    240 241 242 243 244 245 246 247  **  **  ** 251 252 253 254 255


User.Group:               JAMALA.NOGROUP

Startup Program:          RU CI.RUN::PROGRAMS
Working Directory:        ::JAMALA
Logoff Program/Cmndfile:  [LOGOFF.CMD::JAMALA]:
Total CPU Time:           2 HR  30 MIN  10 SEC  209 MSEC
Total Connect Time:       80 HR   6 MIN  23 SEC   48 MSEC
CPU Limit:                No Limit
Connect Time Limit:       No Limit

*****************************************************************
```

Note that LUs 14 and 248 through 250 are not in the LU access table.

New Group (NE G)

Purpose:	Create a new group by creating a new group configuration file and putting it in the /USERS directory with the type extension .GRP.
Syntax:	`NE G`

Description:

You must be a superuser or have the required capability level to create new accounts with the NEW GROUP command.

New accounts may be created during or following system initialization and are usable as soon as they are defined.

The prompts in creating a group are:

```
Enter group logon name:
Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit]:
Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit]:
LU access table modifications ( [–]LU#[:LU#2] ):
```

The following defaults are defined for the group attributes set by the NEW GROUP command:

Attribute	Default Value
Group CPU limit	−1 (no limit).
Group conntect time limit	−1 (no limit).
LU access table	all LUs.

Group Logon Name

This is the group logon name. The name must be unique, follow the file naming conventions, and be one to sixteen characters with no parentheses. The definition is not case sensitive, therefore, either uppercase or lowercase characters can be used.

CPU Usage Limit

This defines the CPU limit for the group. The values for CPU limit are:

hh:*mm*:*ss*	*hh* − hours (up to 5960).
or	*mm* − minutes (up to 59).
hh:*mm*	*ss* − seconds (up to 59).
0	inhibits CPU use by group.
−1	group has unlimited use of CPU.
<cr>	no change to value.

No more members may log on after the CPU limit has been exceeded until you rectify the situation by recording the usage and resetting the usage with the RESET GROUP command or by altering the CPU limit for the group. Users logged on when the limit is exceeded are not affected.

Connect Time Limit

This defines the connect time limit for the group. The values for connect time limit are:

hh:*mm*:*ss* or *hh*:*mm*	*hh* − hours (up to 596500). *mm* − minutes (up to 59). *ss* − seconds (up to 59).
0	inhibits group from logging on.
−1	gives group unlimited connect time.
<cr>	no change to value.

No new members may log on if the connect time limit has been exceeded until you rectify the situation by recording the connect time and resetting it with the RESET GROUP command or by altering the connect time limit for the group. Users logged on when the connect time limit is exceeded are not affected.

LU Access Table

The LU access table defines the LUs to which the group has access. The values for the LU access table are:

n	adds LU *n* to LU access table.
−*n*	removes LU *n* from LU access table.
m:*n*	adds LU *m* to LU *n* to LU access table.
−*m*:*n*	removes LU *m* to LU *n* from LU access table.
`/L`	lists LUs set in LU access table that group can access.
`/E`	ends LU access table modification.
<cr>	ends LU access table modifications if sole input at prompt.

Multiple inputs must be separated by a comma or space. Access to all LUs is the default, remove those that you do not want the group to access. Setting an LU bit that has already been set causes no modification and no warning message is sent.

Example of Group Account Creation

Group has unlimited CPU and connect time and has access to LUs 0, 8, and 24.

```
CI> grump
GRUMP> new group
Enter group logon name : sample
Creating group SAMPLE
Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit]: <cr>
Enter Connect time limit (hh:mm:ss or –1 for No Limit) [No Limit]:
<cr>
LU access table modifications ( [–]LU#[:LU#2] ): –1:23
LU access table modifications ( [–]LU#[:LU#2] ): 8
LU access table modifications ( [–]LU#[:LU#2] ): –25:255
LU access table modifications ( [–]LU#[:LU#2] ): <cr>
GRUMP>
```

LU access table was altered by removing LUs 1 to 23 and 25 to 255 and adding back LU 8. The same modification could have been made with the single string:

```
   LU access table modifications ( [–]LU#[:LU#2] ): –1:23 8 –25:255 /e
```

or at the GRUMP> prompt:

```
   GRUMP> ne g sample –1 –1 –1:23 8 –25:255 /e
```

or with commas marking default values:

```
   GRUMP> new g sample , , –1:23 8 –25:255 /e
```

New User (NE U)

Purpose: Creates a new user configuration file on the /USERS directory.

Syntax: `NE U`

Description:

You must be a superuser or have the required capability level to create new accounts with the NE U command.

New accounts may be created during or following system initialization and are usable as soon as they are defined.

The prompts for creating a user are:

```
    Enter user logon name:
    Enter user's real name [???]:
    Enter password (a <CR> gives no password):
    Enter capability level (31=SU) [10]:
    LU access table modifications ( [–]LU#[:LU#2] ):
    #UDSPs:depth [0:0]:
    Enter the size of the Environment Variable Block in pages [0]:
    Enter working directory name [::name]:
   *Create directory ::name (Yes/No) [N]:
   *What LU should the directory go on [0]:
    Enter the startup command [RU CI.RUN::PROGRAMS]:
    Enter the logoff program/command file [Not Defined]:
    Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit]:
    Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit]:

    Do you wish to include the user in any existing
    group other than NOGROUP (Yes/No) [N]:
   *Enter Group Name (/E or <CR> to end):
   *Should this be the default logon group (Yes/No)[N]:
```

* This prompt is dependent upon the response to the prompt immediately preceding it.

The following defaults are defined for the user attributes set by the NEW USER command:

Attribute	Default Value
User's real name	???.
User password	no password assigned.
Number and depth of UDSPs	0:0 (no UDSPs).
Size of Environment Variable Block	0 (no EVB)
LU access table	all LUs.
Default logon group	NOGROUP.
Working directory	::LogonName.
Startup command	RU CI:PROGRAMS.
Logoff program/command file	undefined.
CPU limit	−1 (no limit).
Connect time limit	−1 (no limit).

Unique User Information

User Logon Name

This is the logon name for the user. The name must be unique, follow the file naming conventions, be one to sixteen characters with no parentheses. The definition is not case sensitive, therefore, either uppercase or lowercase characters can be used.

User's Real Name

This is the user's real name. The real name may be no more than 30 characters. This definition is case sensitive. The name must be enclosed in backquotes (' ') if it is a parameter in a multiple input string. A <cr> gives the default value of "???".

Password

You can define a password for the user. The password can be no more than 14 characters with no commas or blanks. Special characters are not recommended. The default is no password.

Capability Level

This is the user's capability level (USERCPLV). The capability level is an integer value from 0 (lowest) through 31 (highest). A capability level of 31 sets the superuser bit in the user configuration file. A capability level of 0 to 30 clears the superuser bit in the user configuration file. The default capability level is 10. You can only assign capability levels that are equal to or lower than your own.

LU Access Table

The LU access table defines the LUs that can be accessed. The LU access table values are:

n	adds LU *n* to LU access table.
−*n*	removes LU *n* from LU access table.
m:*n*	adds LU *m* to LU *n* to LU access table.
−*m*:*n*	removes LU *m* to LU *n* from LU access table.
/L	lists LUs set in LU access table that group can access.
/E	ends LU access table modification.
<cr>	ends LU access table modifications if sole input at prompt.

Multiple inputs must be separated by a space or comma. GRUMP prompts for LU access table alterations until a /E is entered in the input string or a <cr> is the only input entered at the prompt. The default is access to all LUs. You remove those you do not wish the user to access.

Be sure the user has access to LU 0, used for program-to-program communication. Also make sure the user can access the LUs containing directories such as /PROGRAMS and the default directory needed.

Number and Depth of UDSPs

This defines the number and depth of the User-definable Directory Search Paths (UDSPs) for this user. The number and depth of UDSPs must be separated by a colon with no spaces between the numbers and the colon. The legal values are:

m:*n*	both zero or both not zero (*m* = number, *n* = depth).
m:	modify number only.
:*n*	modify depth only.
<cr>	use default of 0:0.

Environment Variable Block

This is the size of the Environment Variable Block (EVB) in pages, which is allocated to the user's session at logon. The size can be set to any value from 0 pages to 32 pages, inclusive.

USER.GROUP Information

The information required after this point defines associated groups for the user. The information you enter applies first to NOGROUP group and then to any other associated groups.

Default Working Directory

This defines the default working directory for the user when associated with the group. The directory name must not exceed 63 characters including delimiters and must follow directory naming conventions. GRUMP does not verify that the directory belongs to the user whose account is being created.

If the directory specified does not exist, GRUMP prompts to ask whether it is to be created and on which LU it should be created.

```
Create directory XXXXX (Yes/No) [N]:
What LU should the directory go on [0]:
```

The default of 0 creates the working directory on the same LU as your current working directory or the lowest numbered disk LU on which directories can be created. The LU on which the working directory resides must be in the user's LU access table. The second prompt is issued only if the directory should be created.

Startup Command

This defines the startup command (for the user in the group), which is the program to be executed when the user logs on. The LU containing the directory on which the logoff program/command file must be in the user's LU access table. A <cr> leaves the value as it is. The command can have a maximum of 80 characters.

```
Enter the startup command [RU CI.RUN::PROGRAMS]: ru ci.run::programs logon.cmd
```

If the program to be run at logon is RPed with the D option in the Welcome file to increase system performance, it must be specified without a path name at the startup command prompt to force the system to use the proto-ID in XSAM instead of creating one from scratch. Thus, in the above example, the command

```
   ru ci.run logon.cmd
```

should be specified, not "`ru ci.run::programs logon.cmd`".

Starting programs other than CI must be loaded and in the directory specified in the command. A startup command string must be enclosed in back quotes (') if it is a parameter in a multiple parameter string. For example:

```
 GRUMP> alter user jamala.lab , 'ru ci.run::programs logon.cmd' , , 100:0:0
```

Logoff Program/Command File

This defines the logoff program/command file to be scheduled or to which execution is transferred when the user is associated with the group and exits CI. The logoff program must be loaded and on the directory specified. The directory location of any logoff program/command file should also be specified. The LU having the directory where the logoff program/command file is must be in the user's LU access table. If Security/1000 is OFF, the protection of /USERS and the corresponding user configuration file must be RW/R (if groups are not being used) or RW/R/R (if groups are being used) in order for this feature to work.

The program or command file must be enclosed in back quotes (' ') if it is a parameter in a multiple parameter input string. A <cr> at the prompt gives no logoff program/command file.

CPU Usage Limit

This defines the CPU limit for user when associated with the group. The values for CPU limit are:

hh:*mm*:*ss*	*hh* − hours (up to 5960).
or	*mm* − minutes (up to 59).
hh:*mm*	*ss* − seconds (up to 59).
0	inhibits CPU use by group.
−1	group has unlimited use of CPU.
<cr>	no change to value.

The user may not log on in this group after the CPU limit has been exceeded until the you rectify the situation by recording the CPU usage and resetting it with the RESET USER command or by altering the CPU limit for the group. Users logged on when the limit is exceeded are not affected.

Connect Time Limit

This defines the connect time limit for user when associated with the group. The values for connect time limit are:

hh:*mm*:*ss*	*hh* − hours (up to 596500).
or	*mm* − minutes (up to 59).
hh:*mm*	*ss* − seconds (up to 59).
0	inhibits group from logging on.
−1	gives group unlimited connect time.
<cr>	no change to value.

The user may not log on in this group after the connect time limit has been exceeded until you rectify the situation by recording the connect time and resetting it with the RESET USER command or by altering the connect time limit for the group. Users logged on when the connect time limit is exceeded are not affected.

Associated Groups

GRUMP prompts to see if the user should be included in groups other than NOGROUP. If other groups are specified, GRUMP prompts for the same information required for USER.NOGROUP (working directory, startup command, logoff program, CPU and connect time limits). GRUMP prompts for additional group entries and information until a /E or <cr> is entered.

Password (PA)

Purpose: Changes a current user account password.

Syntax: `PA [`*username*`]`

username	User account name that will have its password changed, default is the user's account.

Description:

You must know the present password to change it with this command.

Password modification steps:

```
Present password:
New password:
Retype new password:
```

The password input is not echoed to the terminal.

After verification of the new password, GRUMP changes the password in the configuration file.

Purge Group (PU G)

Purpose: Removes a group from the multiuser account system. The group configuration file is purged from the /USERS directory. The group record and information are removed from the configuration file of all group members.

Caution Do not purge group configuration files from the /USERS directory with the PU command in CI. The PURGE GROUP command in GRUMP purges the group from the multiuser account system with all the necessary cleanup.

Syntax: `PU G` *groupname* `[OK]`

groupname	Group whose account information is to be purged. "@" will purge all groups with the exception of the NOGROUP and SYSTEM groups.
`OK`	Optional parameter to override verification prompt.

Description:

A group account cannot be purged if any user associated with the group is logged on.

GRUMP issues a verification prompt if optional OK parameter is not used.

Purge User (PU U)

Purpose: Removes a user from the multiuser account system. The user configuration file is removed from the /USERS directory. The user record is removed from the members list in all groups in which user is a member.

Caution Do not use the PU command in CI to purge user configuration files from the /USERS directory. The PURGE USER command in GRUMP purges the user from the multiuser account system with all the necessary cleanup.

Syntax: `PU U` *UserGroup* `[OK]`

UserGroup User and group information. Can be specified as follows:

user[.]	User name to be removed from all groups in which a member and user account are to be purged.
user.group	User name to be removed from specified group.
@.group	All members of a specified group to be removed, group account not changed, not allowed for NOGROUP group, MANAGER cannot be removed from SYSTEM group.
user.@	User to be removed from all groups with the exception of NOGROUP, user account not changed, MANAGER cannot be removed from SYSTEM.

`OK` Optional parameter to override the verification prompt.

Description:

A user account cannot be purged if anyone is logged on (associated with any group) as that user.

USER.GROUP account information cannot be purged if the user is logged on associated with the specified group.

GRUMP issues a verification prompt if optional OK parameter is not used.

Reset Group (RE G)

Purpose:	Clears the CPU usage and/or connect time total(s) for all groups or a specific group.
Syntax:	`RE G` *group* `[CP|CO]`

group	One or more groups whose totals will be reset, “@” means all group totals are to be reset.
`CP`	Optional parameter that resets only the CPU usage for the specified group.
`CO`	Optional parameter that resets only the connect time total for the specified group.

Description:

Resetting a group’s totals does not affect the individual USER.GROUP totals for group members. User totals can be reset with the RESET USER command.

If neither the CP nor CO parameter is specified, both totals are reset. CP and CO cannot be specified at the same time.

Reset User (RE U)

Purpose:	Clears the CPU usage and connect time totals for a specific user (grand totals), a single user within a group (USER.GROUP totals), all users within a group (USER.GROUP totals), or a user within all groups where user is a member (USER.GROUP totals).
Syntax:	`RE U` *UserGroup* `[CP|CO]`

UserGroup User and group information. Can be specified as follows:

user	Account totals unique to the user and appropriate totals in *user*.NOGROUP record for systems not using groups. *User.group* totals are not affected.
user.	Cumulative totals for *user* (sum of all *user.group* totals).
user.group	*User.group* totals for *user* within specified group.
@.group	*User.group* totals of all the members of the group when associated with the specified group at logon. GRUMP loops through the member lists and resets member totals one at a time. The cumulative totals for the user or group are not affected.
user.@	*User.group* totals for the user in each group where the user is a member. GRUMP loops through the group list and resets total in the user.group records one at a time. The cumulative totals for the user are not affected.

`CP`	Optional parameter to reset only CPU total for the specified user.
`CO`	Optional parameter to reset only CONNECT total for specified user.

Description:

If neither the CP or CO parameter is specified, both are reset.

Run (RU)

Purpose: Clones and runs a program.

Syntax: `RU` *program* [*ParmString*]

program	Name of the program to run.
ParmString	Parameters for the listed program.

Transfer (TR)

Purpose: Allows the GRUMP user to alternate between interactive and non-interactive command input modes.

Syntax: `[/]TR` *InputSource* | −*n*

InputSource	Terminal LU or file name from which commands are read.
−*n*	Negative integer that causes the control input to go back *n* levels.

Description:

TR and /TR are identical except that /TR can be issued from within GRUMP commands.

Transfer commands can be nested to a depth of 5 levels. For example, a TR command issued at a terminal transfers control to a command file that contains a command transferring control to a terminal LU. At this point, transfers are nested 2 deep.

The negative integer (−*n*) transfers control back n levels. If *n* is greater than the number of nested levels, all previously nested input sources are closed and GRUMP goes into interactive command input mode at the user's terminal.

An end-of-file condition is interpreted as "/TR −1" and transfers control to the previous level.

If an irrecoverable error is encountered in a command file, GRUMP transfers control to the terminal, flushes the input string, and reprompts the user. A subsequent "/TR −1" transfers control back to the command file where the error occurred.

Hitting the BREAK key forces transfer to the terminal. A "/TR −1" transfers control back to the command file that is in effect at break.

TRANSFER commands are written to the log file:

```
* GRUMP>
* >> TR
* Enter transfer file of -(# of levels back) :
* >> filename/LU
* GRUMP>
```

An asterisk in the first column changes the commands into comments that are not executed if GRUMP is re-run using the log file as an input source.

The ">>" highlights the transfer command and the source of subsequent commands.

A command file exited with an implied return, EOF, appears as:

```
* GRUMP>
p* GRUMP>
```

A transfer command used to exit a command file appears the same as shown in the preceding log file example.

SESLU Utility

SESLU modifies and lists session LU access tables. It does not affect the user or group configuration file associated with the session user. If a session user is given access to an LU, the corresponding bit is set in the session LU access table. If a session user's access to an LU is removed, the corresponding bit in the session LU access table is cleared. You can never remove access to the terminal LU of a session or to an LU in the UDSPs of a session.

Calling SESLU

To call SESLU, enter the following runstring:

```
   CI> [RU] SESLU [+S:session] [[-]lu[:lu]] [[-]lu[:lu]] [[-]lu[:lu]]
```

+S:*session* is the session number whose LU access table is to be modified or listed. If you do not specify the number, SESLU defaults to the caller's session.

You may enter a positive or negative [-]*lu*[:*lu*] parameter. A positive entry indicates the LU number or range of LUs to set in the session LU access table. A negative entry specifies the LU number or range of LUs to clear in the session LU access table. If you do not enter any LU number or range, the session LU access table is listed.

Loading SESLU

To load SESLU, use the following LINK command:

```
   CI> link,seslu.lod
```

SESLU Protection

If Security/1000 is turned on, the system manager can assign the different capability levels required to perform all the SESLU subfunctions. These include listing and modifying your session's and another session's LU access tables.

If the Security/1000 system is not turned on, any user can list a session LU access table with SESLU, but only superusers can modify a session LU access table.

Returned Values

SESLU returns the following values through a PRTN call:

Word 1	Status (0 = successful, -1 = unsuccessful).
Words 2-5	0, not used.

SESLU Examples

Example 1: List the LU access table for session 102.

```
CI> seslu +s:102
```

Example 2: Add LU 1 to the LU access table for session 102.

```
CI> seslu +s:102 1
```

Example 3: Add LUs 1-32 and 34-50 to the caller's LU access table.

```
CI> seslu 1:50 -33
```

Example 4: Remove LUs 44-60 and add LUs 20-30 to the caller's LU access table.

```
CI> seslu -44:60 20:30
```

Example 5: Add LUs 20-90 to the LU access table for session 116.

```
CI> seslu +s:116 20:90
```

Example 6: Add LUs 3 and 5 and remove LUs 2 and 4 from the caller's LU access table.

```
CI> seslu -2 3 -4 5
```

KILLSES Utility

KILLSES terminates a session immediately. The session is logged off, all programs associated with that session are killed, associated spool files are closed and released, and the user ID entry in the User Table for that session is released. The session can be any type: background, interactive, or programmatic. The system session, session 0, cannot be killed. KILLSES cannot be used to kill a session that is also running KILLSES.

Calling KILLSES

To call KILLSES, enter the following runstring:

```
   CI> [RU] KILLSES [sessionnumber] [OK]
```

The *sessionnumber* parameter is the number of the session to be killed, if specified. If not specified, you are prompted for the number of the session to be killed.

When you select the OK parameter, KILLSES is executed immediately, without prompting you for verification.

Loading KILLSES

To load KILLSES, use the following LINK command:

```
   CI> link.killses.lod
```

KILLSES Protection

If Security/1000 is turned on, the system manager can assign the capability level required to run KILLSES; otherwise, only a superuser can run KILLSES.

Returned Values

KILLSES returns the following values through a PRTN call:

Word 1	Status (0 = successful, -1 = unsuccessful).
Word 2	If word 1 = 0, the number of the session that was killed. If word 1 = -1, this is not used and will be 0.
Words 3-5	0, not used.

KILLSES Examples

Example 1: Terminate session 160 immediately.

```
CI> killses 160 ok
```

Example 2: Terminate session 150 after user verification.

```
CI> killses 150
```

Example 3: Prompt the user for session number and verification.

```
CI> killses
```

4

File and System Security

Security in the RTE-A system consists of two separate but overlapping subsystems, File System Security and Security/1000 (see Figure 4-1).

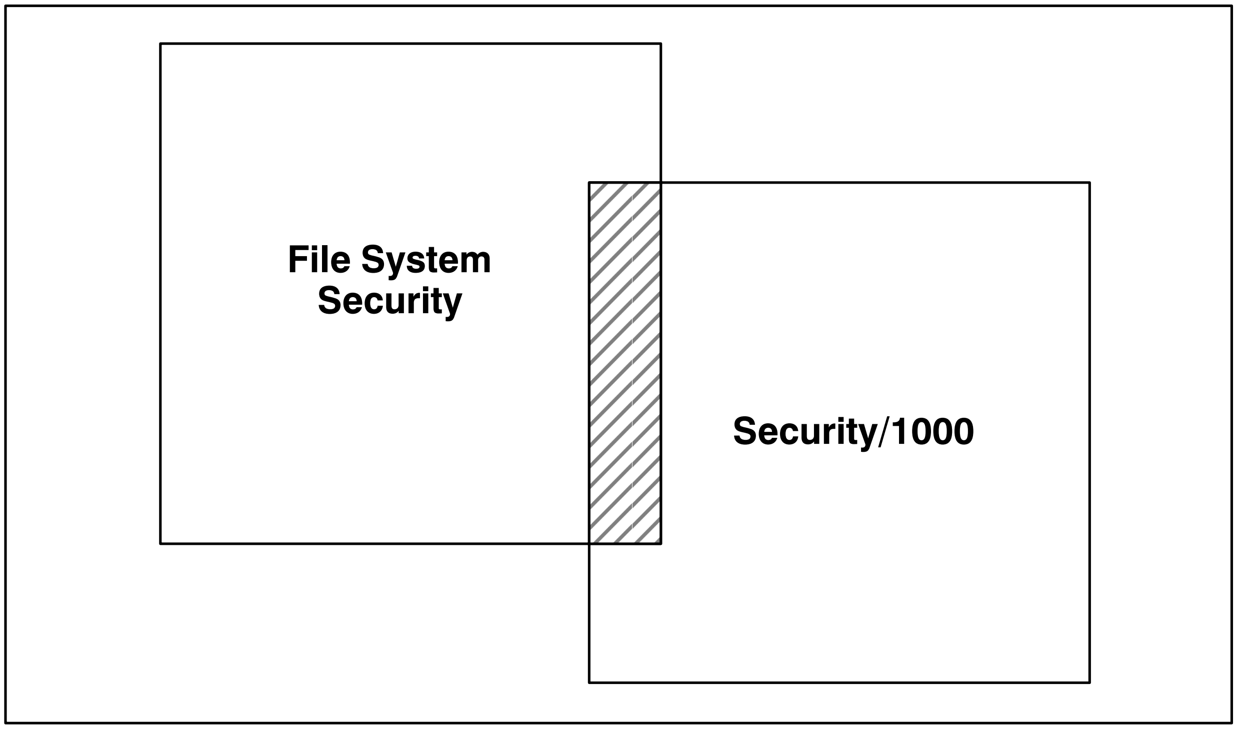

Figure 4-1. RTE-A Resource Protection

Both subsystems are built into the multiuser environment, but Security/1000 need not be turned on if you, the system manager, do not want this checking mechanism. File System Security, including LU access tables, operates whether or not Security/1000 is turned on.

File system security and Security/1000 overlap in the area of file system access. For example, if a user tries to run a program that is not RPed, the user must first have read access to the program file in order to RP the program (file system read/write protection) and then have the capability to run the program (Security/1000 program access protection). If either check fails, the request to run the program is denied. Another example is with the CL (Cartridge List) command. A session user may have the capability to execute the command, but the command does not list any LUs that are not in the session LU access table (file system LU access protection).

The concept of superuser (CPLV 31) has a different effect on the file system RW (Read/Write) access protection and the LU access protection. If a user is a superuser, the file system RW access protection is overridden, but the LU access protection is not. In the first example, superusers are able to run the program even if they do not have read access to the program file. In the second example, only LUs that are in the LU access table are listed.

As system manager, you must decide whether or not Security/1000 is to be turned on, and if so, which Security/1000 configuration will be most applicable to your system.

File System Security

File System Security provides file, directory, and volume, ownership and protection. Under File System Security, users are divided into superusers and non-superusers. Superusers have a user capability level of 31 and are not subject to file system protection checks. Non-superusers may be assigned user capability levels from zero to 30, inclusive, and are subject to file system protection checks. As the system manager, you are a superuser, and usually the only superuser. It is your responsibility to assign user capability levels.

As system manager, or superuser, you have full system access including read/write access to any file (program as well as text), directory, and volume on the system. You are the original owner of all system directories but can re-assign this ownership and associated protection privileges.

Security Under CI

In the CI, or hierarchical file system, each directory has an owner and an associated group. All files in a directory are owned by the directory owner and have the same associated group as the directory. The owner can change the protection status (defined as read/write access allowed the owner, members of the associated group, and general users) of the files in the directory.

The session user who creates a directory is the initial owner of that directory, and the directory has the owner's associated group. The owner can re-assign directory ownership and associated group. The owner and associated group of a subdirectory can be different from the owner and associated group of the directory containing the subdirectory.

File Ownership and Protection

File protection, a security measure in the CI file system, is defined when a file is created or copied into a directory. It can be specified differently for the owner, for members of the associated group, and for general users. The default protection is to allow the owner both read and write access, and associated group members and general users only read access. Protection can be specified on a file-by-file basis in any combination of read and write access for the owner, for members of the associated group, and for general users.

Directory Ownership and Protection

Directories have protection information that is slightly different from that of files. The protection status of a directory applies to any file, or subdirectory, created in that directory. Users cannot use CI commands to change information in write-protected directories (read only access specified). They cannot create, purge, or rename files or subdirectories in read only directories. Users cannot look at the contents of directories that have read protection.

Volume Ownership and Protection

Like a directory, an entire CI volume can have an owner and associated group. The initial owner of a volume is the session user initializing the volume. The associated group for the volume is the associated group of the owner. Ownership and associated group may be reassigned. Volumes also have protection information. The protection status of a volume regulates the reading, creating, purging, or renaming of global directories on that volume. Note that the owner of a global directory can be different from the owner of the volume on which it resides.

In CI, any directory may access remaining volume space. If you want to restrict disk space usage, you can assign each CI volume an owner who will set volume-wide protection. Use the luV parameter in the CI OWNER and PROT commands to display and modify volume ownership and protection. You can also use FMP routines (FmpOwner, FmpProtection, FmpSetOwner, and FmpSetProtection) to retrieve and modify volume ownership and protection programmatically.

LU access tables provide another means of limiting access to CI volumes. If the disk LU identifying a volume is in a user's session LU access table, access to the volume is granted. If the LU is not in the table, all access to the volume is denied.

Security under FMGR

In the FMGR (File Manager) file system, protection is defined by accessibility. FMGR cartridges (a particular LU defined at system generation) have a single directory and a single set of files.

Security/1000 (VC+ Only)

Security/1000, part of VC+ (HP product number 92078A), is a set of building blocks for software developers that enables them to create a secure application environment. It is NOT meant to set up a secure system that has program development going on. It is designed to be compatible with existing file system protection, have minimal impact on real-time processes, and be an integrated part of resource management. It is:

- Switchable—can be turned on and off by the superuser with a single command.
- User configurable—can be tailored by the superuser by modifying the supplied template.
- User extensible—can be secured by the superuser to create a unified security environment.
- Table-driven—can be configured and modified by the superuser while the system is running.

Security/1000 consists of capability levels to classify users, system security tables that contain the rules that govern the user's activities, and interfaces to the security system. Applications or utilities must be written to implement Security/1000 so that security checks on a user's ability to execute a command can be made.

System Manager Responsibilities

If Security/1000 is used, your responsibilities as system manager include:

- Determining user, command, and program capability levels within the multiuser environment.
- Installing the subsystem at system generation or regeneration.
- Defining a default logon account for DS transparency software to use when a request comes from another node without specifying an account name.
- Using the SECTL utility to initialize and turn on Security/1000.
- Maintaining the security tables online with SECTL.
- Altering security tables offline, running SECTL GT, and regenerating (to add more categories and/or category functions).
- Maintaining the subsystem using the GRUMP utility to alter user capability levels, and using SECTL and LINK to modify program capability levels.

Note If a default logon account is not defined when a DS transparency request arrives, the default TRFAS SESSION is used for all file accesses. Since TRFAS SESSION has full access to a system with Security/1000 turned ON, it is VERY IMPORTANT that the default logon account be defined.

The default logon account is created using GRUMP. It defines the environment in which all file accesses for requests using the default DS account are performed. The default account is defined in the DS system using the DS monitor DSRTR. (See Chapter 9 in the *System Generation and Installation Manual* for details.)

Capability Levels

A capability level (CPLV) is an integer in the 0 through 31 (inclusive) range. You, the system manager, can assign all users capability levels (USERCPLV), which are stored in the user configuration file and copied to the user ID table entry when the user is logged on.

You also can assign a capability level to all CI and system level commands (CMDCPLV). The CMDCPLVs for all commands are kept in security tables maintained by SECTL. If a command does not appear in a security table, it has a default capability of zero where no restrictions apply to the command. Note that a set of fully configured tables (SECURITY.TBL) is supplied with the product and shown in Appendix D.

All programs have three capability levels; two are accessible via SECTL and the Security/1000 routines and the third is for Security/1000 internal use. All three capability levels are stored in the program's ID segment extension. These CPLVs are:

PROGCPLV	Program capability level. PROGCPLV is the limiting factor on what functions a program can perform. PROGCPLV must be equal to or greater than function CPLV. PROGCPLV is accessible via SECTL and Security/1000 routines.
RQUSCPLV	Required user capability level. USERCPLV must be equal to or greater than RQUSCPLV in order to run the program. RQUSCPLV is accessible via SECTL and Security/1000 routines.
ORGCPLV	Original capability level. ORGCPLV is a copy of the original PROGCPLV given to the program at link time or set with the SECTL utility. ORGCPLV is for Security/1000 internal use; it is not directly accessible to users.

All fields of $CPLV have a default of zero.

You need to be aware of user, program, and system capability requirements as well as File System Security when determining capability levels. If the functions that users need have been assigned CPLVs higher than their own, users are not able to continue without some indirect means of accessing those programs (see the "Program Access Protection" section).

When a program is run with Security/1000 turned on, PROGCPLV is assigned the greater value between the session user's USERCPLV and the program's original PROGCPLV. Programs assigned a low CPLV can therefore perform tasks requiring a higher CPLV if the user requesting them has that higher CPLV. In the example in Figure 4-2, a program is linked with an original PROGCPLV of 20 and a RQUSCPLV of 15. When the program is scheduled by a user with a USERCPLV of 25, it is assigned a PROGCPLV of 25 because the USERCPLV is greater than the original PROGCPLV. When the program is scheduled by a user with a USERCPLV of 17, it is assigned a PROGCPLV of 20 because the original PROGCPLV is greater than the USERCPLV.

If a program is scheduled by another program, the USERCPLV of the session in which the parent program is running is checked against the RQUSCPLV of the child program to determine whether the scheduling program can run the child program.

Security Tables

The security tables reside in the system and are used by Security/1000 to govern function and resource accessibility. They consist of a system manager defined set of rules based on capability levels, categories, and functions. The lables are created using STGEN or the SECTL GT command with a security table source code module. The output of STGEN and SECTL GT command is an OS module that is generated into the system. Appendix D provides a description of the source format of the security tables and a sample of the source code of the SECURITY.TBL module that is shipped with the VC+ software. Appendix E provides a listing of the structure of the security tables, as well as a worksheet to aid in the creation of the security tables. Appendix F provides an interpretation of the security errors. Security/1000 library routines are listed in Appendix G.

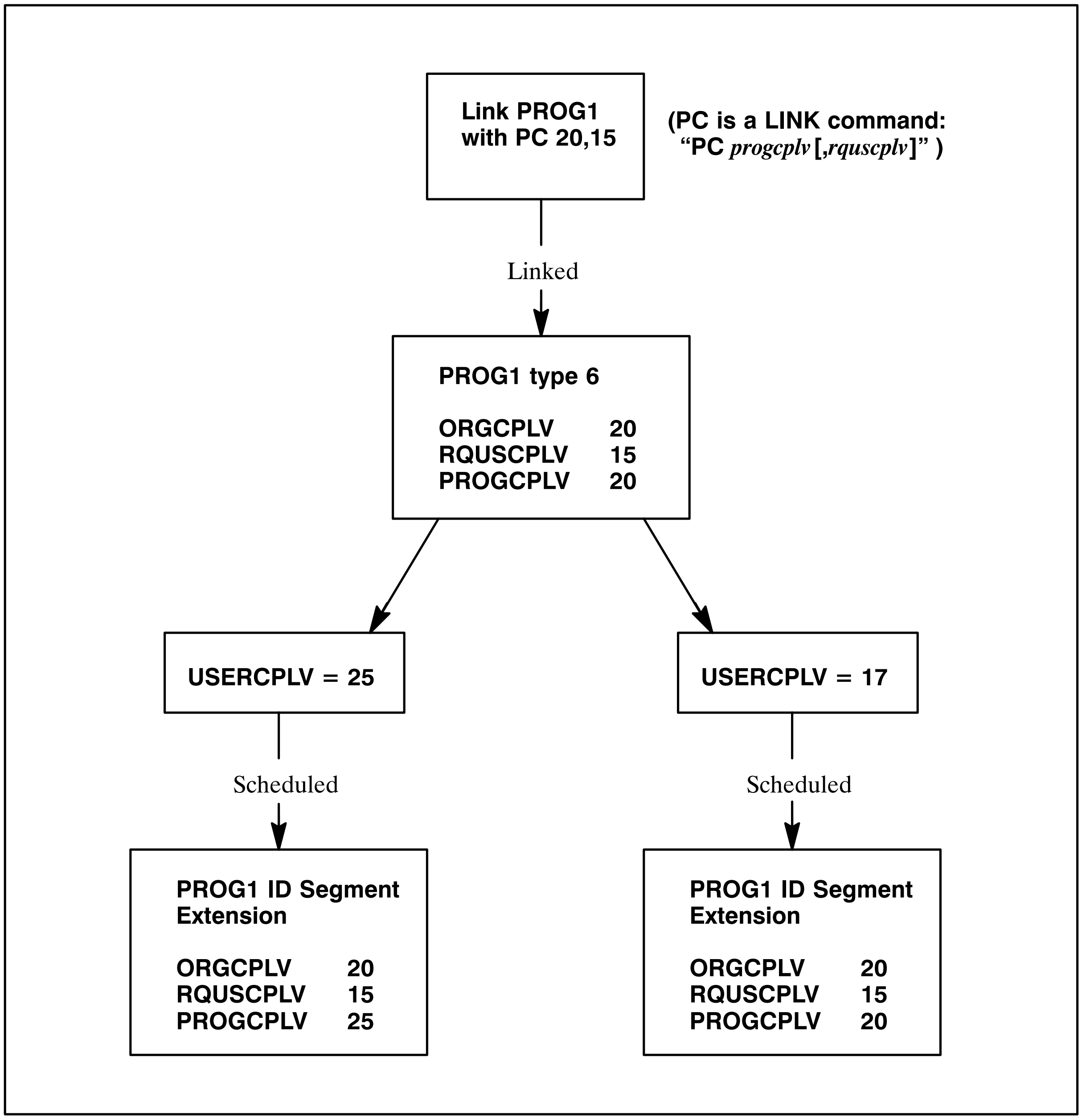

Figure 4-2. Example of Program Security Assignment

Utilities with Security/1000 Implemented

The HP utilities that have an implemented Security/1000 internal security check are:

- CI (Command Interpreter): RTE-A hierarchical file system user interface.
- GRUMP (GRoup and User Management Program): Command-driven utility for managing a multiuser account system.
- SECTL (SECurity TabLe): Utility for performing all Security/1000 control functions.
- LINK: Linkage editor for RTE-A operating system.
- KILLSES (KILL SESsion): Utility that terminates a user session with RTE-A.
- SESLU (SESsion Logical Unit): Utility that lists and modifies session LU access tables.

Other utilities may inherit security checking by calling routines that, within themselves, have implemented security checks. As an example, LI does not perform security checking within itself but it inherits it when calling lower level routines that have security checking implemented (for example, FMPOPEN with the CREATE and OPEN options).

All utilities (HP and user-written) and sets of routines (for example, FMP) that implement some level of internal security checking are listed as categories (described below) in the system security tables (see Appendix D).

Forms of Security Implementation

There are a variety of forms in which security is implemented. For example, GRUMP has multiple CPLV levels defined for each command. CI, on the other hand, defines a single CPLV per command on a single-level basis. By examining the function definitions for CN and CD (in the CI category) in the default security tables, you can see that users must have a CPLV of 10 or greater to use CN, while a user having *any* CPLV can use CD. The rationale behind the absence of a specific CPLV for CD is that CD is protected by security checks in lower level FMP routines and the file system. The fact that both CD and CN have entries in the security table, and CI performs security checks on a single level for all CI commands, gives the system manager the ability to define the CPLVs as desired.

The LINK program is yet another example of differing forms of security implementation. It can be seen, by examining the security tables, that only five of its commands (OS, PR, LC, SC, and SH) have security implemented. This form of implementation provides the system manager with the ability to change the CPLVs on these five commands as they have been defined. CPLVs, however, cannot be assigned to other LINK commands by adding them to the security table.

Categories and Functions

A category is a group of functions such as the CI commands or the FMP routines. The category designer selects the functions for each category.

A function is any activity carried out by the system on behalf of the user. An individual function can be divided into four levels, the base function and up to three subfunctions. For example, the CI command TM displays the system time as its primary or base function. It also has the subfunction of setting the system clock. The function designer determines the base and subfunctions of any function.

The Category Index Table (CIT) and the Category Function Tables (CFT) are the security tables that determine access to system resources. See Figure 4-3 for the basic structure of these tables. The size of the CIT and CFTs are limited by available memory.

The CIT contains an entry for each category including its name and address. The CIT determines which CFT is searched for a given category. The CFTs contain an entry for each function of a particular category. There is one CFT for each entry in the CIT. A CFT entry holds up to four integer values per function. These values are usually capability levels.

When Security/1000 checks to determine whether or not a function may be performed, it first searches the CIT for the address of the category's function table. Then it searches that particular CFT for capability information about the function and allows or disallows the function accordingly.

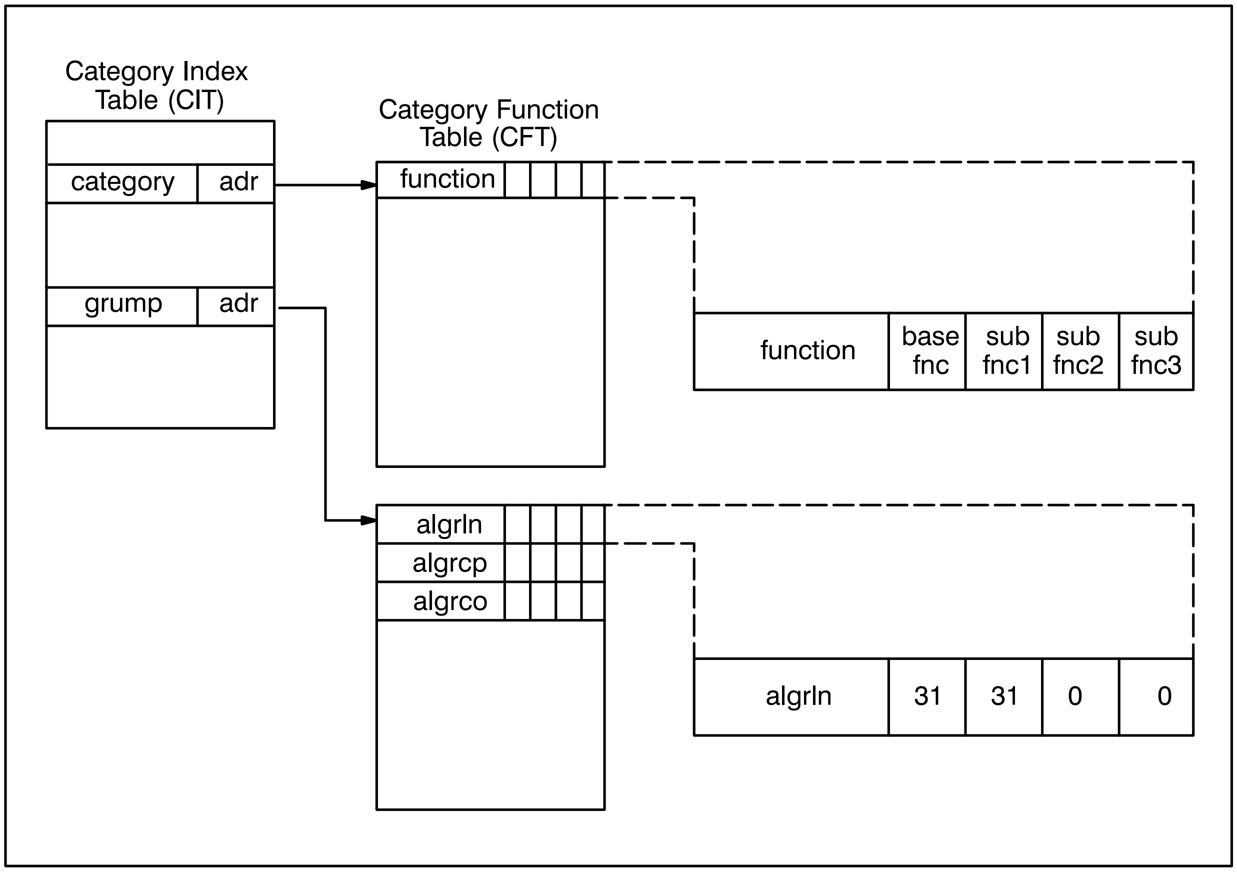

Figure 4-3. Security Table Structure

Searching the CIT and CFTs is done serially. If you have a category with a large number of functions, you derive a performance advantage in having the functions divided between two smaller categories rather than having them in one large category.

Security Table Format

The security tables are accessed via the $VCTR entry point $SEC.PNTR that points to the first word of the CIT. The first word of the CIT is the number of entries in the CIT. Each entry of the CIT has the following format:

Word	Contents
1-3	The category identifier. This is an upshifted, left-justified, and blank-filled ASCII string that uniquely identifies the category to Security/1000. The category name can be any length but only the first six bytes are put in the entry. Therefore, the name must be unique within the first six bytes.
4	The address of the CFT for the category identified in words 1 through 3.

The first word of the CFT is the number of entries in the CFT. Each entry of the CFT has the following format:

Word	Contents
1-3	A function, left-justified, upshifted, and blank-filled. The function can be any length but only the first six bytes are put in the entry. Therefore, all functions for a given category must be unique within the first six bytes.
4	The Base CPLV for the function in the range 0 through 31 inclusive.
5	Subfunction 1 CPLV in the range 0 through 31 inclusive.
6	Subfunction 2 CPLV in the range 0 through 31 inclusive.
7	Subfunction 3 CPLV in the range 0 through 31 inclusive.

The values in words 4 through 7 can be greater than 31 if the function holds non-security information.

Modifying Security Tables

The two methods of modifying the security tables are:

1. Creating a new set of tables by:
   a. Altering a copy of the security table source file (the original SECURITY.TBL file or your already modified security table);
   b. Using the SECTL GT command (or STGEN) to generate a new set of tables;
   c. Regenerating the system; and
   d. Rebooting.

2. Modifying online security tables while the system is running by:

   a. Using SECTL commands to alter existing security tables, or

   b. Using the SEC1000.LIB subroutines (the programmatic interface of Security/1000) described in Appendix G to alter existing security tables.

   This method modifies the tables in memory, but not the swap files, so any online modifications are lost when the system is rebooted. Since you decide the definition of values, you can define non-security information as values. You can, for example, design an application that sends reports to different LUs.

Security Information in Security Tables

Security tables primarily hold security information. In the security table source file example below, the function defined holds only security information.

```
Category: GRUMP
 ALGRLN 25 31 0 0
```

The category is the GRUMP utility for managing resources in the multiuser system. The function is defined as altering a group logon name. The base function, altering the group logon name of a group the caller belongs to, requires a capability of 25. Subfunction 1, altering the group logon name for a group the caller does not belong to, requires a capability of 31. Subfunctions 2 and 3 are not defined.

The code in GRUMP to make use of the ALGRLN function in category GRUMP might look like the following:

```
integer*2  CPLVtoCheck,OperatorCPLV,ProgCPLV,RqusCPLV,OrgCPLV
integer*2 error,flags(4)
  .
  .
  .
if (caller is a member of group being altered) then
    CPLVtoCheck = 1
else
    CPLVtoCheck = 2
endif
call  SecGetCPLVs  (OperatorCPLV,ProgCPLV,RqusCPLV,OrgCPLV)
call  SecChkCPLVNam  (6HALGRLN,6HGRUMP  ,OperatorCPLV,flags,error)
if (error.lt.0) call ReportError ('Error in performing CPLV check')
if (flags (CPLVtoCheck) .lt.0) call ReportEror ('Insufficient CPLV')
  .
  .
  .
```

Non-Security Information in Security Tables

A CFT can also hold non-security information. The interpretation of values depends upon their definition in the program that uses them.

Since you decide the definition of values, you can define non-security information as values. You can, for example, design an application that sends reports to different LUs.

```
Category: REPORT
  REP01  8  0  0  0
  REP02 10 11 12 13
```

In this example, the integer 8 is defined as the LU to which REP01 is sent and integers 10, 11, 12, and 13 are defined as LU numbers to which REP02 is sent. Also, zero is used as a place holder. Note that the application processing the values is responsible for interpreting their meanings.

The code to retrieve the information for REP01 might look like the following:

```
integer  entry(7),num,addr,error
  .
  .
  .
call SecGetCftNam (6HREP01 ,6HREPORT,entry,num,addr,error)
if (error .lt.0)Call ReportError (error)
LUToSendReportTo = entry (1)
  .
  .
  .
```

If a function requires more than four values, a way to work around the limitation of four values per function is to define two functions identically. Expanding on the previous example,

```
Category: REPORT
  REP01  8  0  0  0
  REP02 10 11 12 13
  REP2a 14  0  0  0
```

If the application has defined REP2A as identical to REP02, the integer 14 is interpreted as a fifth LU to which REP02 is sent.

Another example of the way in which function designers define values to accomplish their purpose is using the rangeoff option when defining functions to specify values that are outside the 0 to 31 range. In the following example, the enable function of the application is set to 40, a value outside the 0 to 31 (inclusive) range:

```
Category: APPLIC
   ENABLE 40 0 0 0
```

The integer 40 is defined as the enabling value and any other value would disable the application. The base function is set to 40 to enable, a number other than 40 to disable.

Security/1000 Interfaces

Security/1000 has two interfaces, the SECTL and STGEN utilities that are the user or interactive interface, and the SEC1000.LIB subroutine library that is the programmatic interface. SECTL and STGEN are described in Chapter 5, and SEC1000.LIB is in Appendix G.

See Appendix H for examples of two programs using security routines. The first program renames or edits the security tables. The second program subjects a user to a series of security checks based on the security tables before it creates directories. Programs such as CI, GRUMP, and LINK perform similar security checks on their commands.

Program Access Protection

Because programs have three CPLVs, you can design applications whose capability is higher than that of the user. Users can be denied direct access to potentially dangerous functions, but be allowed access indirectly through the controlled environment of the application or utility.

There are two levels of checking before a user can run a program:

Level 1 File system security. A user must be able to access a program for it to be RPed. It must be RPed in order to be run.

Level 2 Capability level. The USERCPLV of the session that makes the scheduled request must be greater than or equal to the RQUSCPLV of the program being scheduled. This check is made by the RTE-A scheduler.

If a program does not have an ID segment, both levels of security checking must be passed to run the program. If the program has an ID segment and is RPed, only the second level must be passed because file system security is not involved with the schedule request.

If a user passes the first level and RPs the program but fails the second level and cannot schedule it, the operating system removes the ID segment and returns a message to the user's terminal. It is possible that a user able to run a program if it were RPed is prevented from running it because of file system security.

Programs in the system session are handled in the same way as programs in a user session. However, as the system session always has a USERCPLV of 31, programs running in it behave as though run by a superuser.

The two examples that follow illustrate program access within a secured environment.

Example of a Program Access Utility

Assume that a system has these user categories with these CPLVs:

General users	5
Operator 1—MAX	10
Operator 2—JAMALA	12
Application programmers	20
System programmers/System managers	31

Also, assume that operator 1, MAX, has the job of removing old files from a data base that receives new files daily (see Figure 4-4). MAX has a USERCPLV of 10. Because the Purge Data File function has a CPLV of 26, Security/1000 prevents MAX from removing the old files. Therefore, MAX's system manager created a utility, UTIL1, that allows specific, limited access to the Purge Data File function.

UTIL1 specifies that only files input at least two weeks prior to current system time may be removed. It has a RQUSCPLV of 10 so that MAX can access it, and a PROGCPLV of 26 so that the program can access the purge function. UTIL1 affects only users with CPLVs of 10 through 25. Users with CPLVs below 10 are still denied access and users with CPLVs greater than 26 have full access to the purge function anyway. UTIL1 provides limited (CPLVs 10-25) and specific (purge only selected dated files) access.

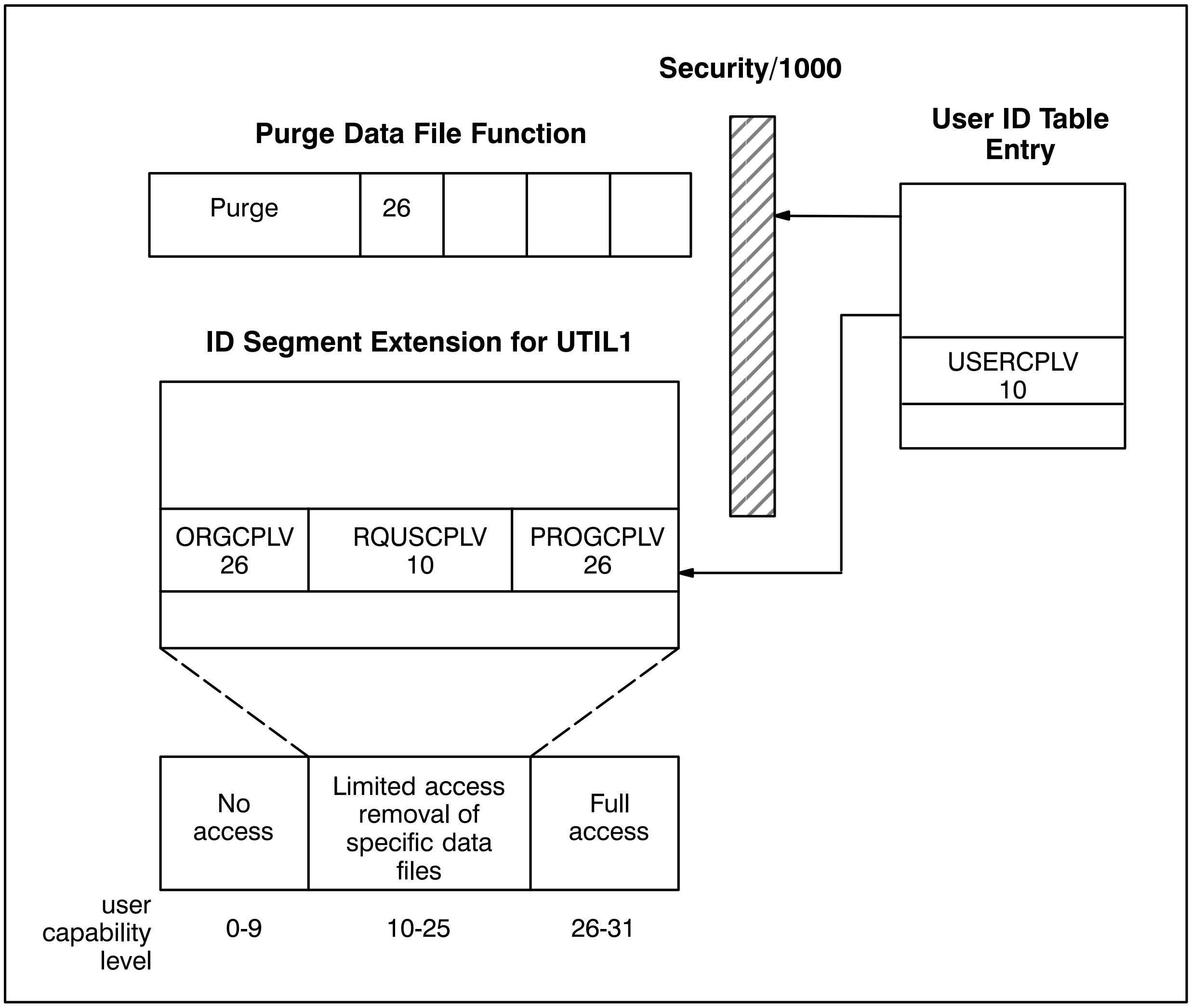

Figure 4-4. Example of Program Access Utility

Example of a CPLV Modification Program

Assume that it is the job of operator 2, JAMALA, to do backups and restores via BACKUP (see Figure 4-5). JAMALA has a USERCPLV of 12. This example assumes there is a user application, BACKUP, that the system manager has used Security/1000 routines to modify so that it has different capability levels defined to access different functions within the utility.

At installation, the section of the security tables relating to BACKUP would be as follows:

```
$fmp:
              .
              .
              .
              crdir 20 20 20 20
              .
              .
              .
 $category: backup
*
* < Function cos is defined as storing files to tape
* < and the subfunctions define what copy options can be
* < used with the base function.
* < base fnc: who can store files to tape
* < subfnc01: who can use the verify option
* < subfnc02: who can use the purge option
* < subfnc03: who can use the append option
*
              cos 10 10 25 12
*
* < Function cor is defined as restoring files from tape
* < and the subfunctions define what copy options can be
* < used with the base function.
* < base fnc: who can restore files from tape
* < subfnc01: who can use the verify option
* < subfnc02: who can use the replace duplicate option
* < subfnc03: who can restore files to directories other than where
* <           they originated
*
              cor 12 10 12 31
*
$category: bkupus
              user01 1
              .
              .
              .
              user10 10
              user11 11
              user12 12
              .
              .
              .
              user31 31
*
```

BACKUP was linked with a PROGCPLV of 20 and a RQUSCPLV of 10.

When JAMALA runs BACKUP, the system sets PROGCPLV to 20, which is the greater value of PROGCPLV (20) and USERCPLV (12). This lets JAMALA restore files from a tape requiring that BACKUP create directories even though the directory creation function has a CPLV of 20 (which is beyond JAMALA's CPLV).

BACKUP was linked with a PROGCPLV of 20 but the security table listed above sets the CPLV of subfunction 2 of the cos function to 25. As a result, only users with a USERCPLV equal to or greater than 25 can access the purge option.

When MAX runs BACKUP, the PROGCPLV is 20, which is the greater value of PROGCPLV (20) and USERCPLV (10). This is not a desired result because it gives MAX access to a function (cor) and an option (append of the cos function) to which he should not have access. MAX has access to the same base and options of these functions as JAMALA.

To avoid this situation, BACKUP has a dynamic PROGCPLV modification facility implemented with Security/1000 routines and a security table category called BKUPUS. This PROGCPLV modification facility causes BACKUP to raise or lower its PROGCPLV during execution. It can lower its PROGCPLV to zero and raise it to the maximum of USERCPLV and ORGCPLV.

Now when MAX runs BACKUP, the modification facility detects the USERCPLV of 10 and lowers its PROGCPLV to 10. When JAMALA runs BACKUP, the modification facility changes the PROGCPLV to 12; therefore, MAX is not allowed to use the restore function but JAMALA is. When JAMALA is performing a restore, BACKUP raises its PROGCPLV to 20 prior to the creation of the directory and lower it to 12 after the directory is created.

Category BKUPUS is created with SECTL or Security/1000 routines if there are extra categories in the current security table (see the Installing Security/1000 section). If there are no extra categories in the current security table, you must create a new table that includes BACKUPUSER and generate the tables in your system. These values are easier to change if need arises because they are in the security tables rather than coded into the program.

BACKUP determines the correct value for its PROGCPLV by means of BKUPUS, which contains the values to which PROGCPLV is mapped for a given user. BACKUP concatenates USER and the ASCII representation of the USERCPLV to get the function name (USER10 for MAX and USER12 for JAMALA in this example). It then calls routine SecGetCftNam to get the value to which its PROGCPLV should be mapped. Note that while the function value is equal to that of the USERCPLV in this example, it does not have to be.

BACKUP calls routine SecChangeCplv to set its PROGCPLV to the correct value. Once the correct value for its PROGCPLV has been set, no special internal filtering by the program is required to restrict a low CPLV user to the desired functionality subset. The PROGCPLV modification facility causes BACKUP to raise its PROGCPLV to the required level before attempting to create a directory for the restore operation. After the directory is created, it then resets PROGCPLV to the value obtained from BKUPUS. All filtering is done by the Security/1000 routines.

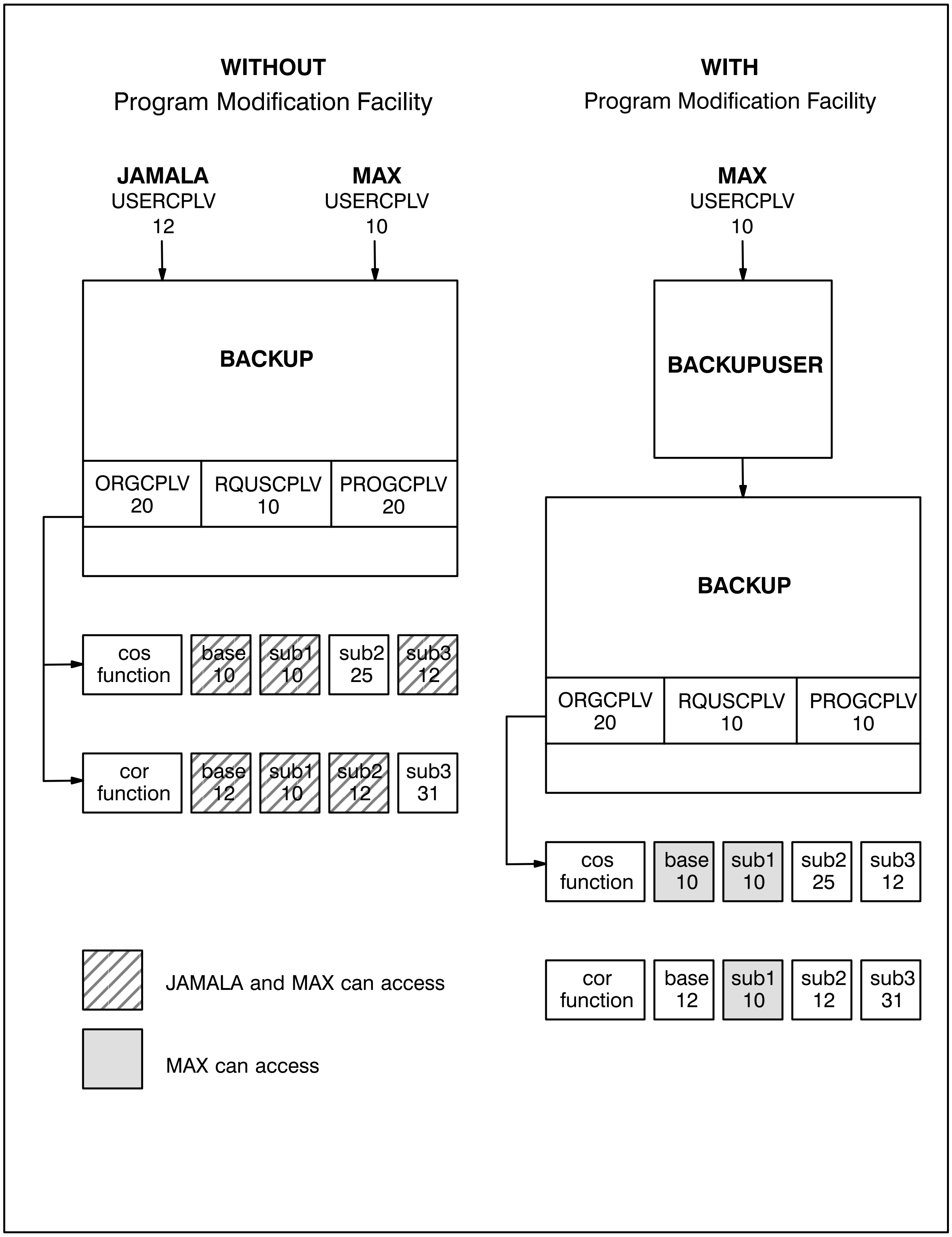

Figure 4-5. Example of a Program Modification Facility

Installing Security/1000

To install Security/1000, proceed as follows:

1. Load Security/1000 software (SECTL.LOD and STGEN.LOD) onto an RTE-A system with Revision 5000 or later.

2. Generate a set of tables using the SECTL GT command. See the GT command in the SECTL utility section in this chapter.

   A sample SECURITY.TBL (source code module) and SECURITY.REL are shipped with the product. If you do not want to alter the security tables and you do not want a SECURITY.MAC module, you do not need to generate a new set of tables. You can use the shipped SECURITY.REL in your system.

   If you do not want to alter the tables, but you want a SECURITY.MAC module, use the SECTL GT command (or STGEN) and SECURITY.TBL to create modules SECURITY.MAC and SECURITY.REL.

   If you want to alter the security tables to add categories or functions, you must alter a copy of SECURITY.TBL or your current security table source file. Use the SECTL GT command (or STGEN) and the modified security table source module to generate a new set of tables. Note that the other SECTL commands are used to modify and list categories and functions that have already been generated into a running system.

   To avoid having to generate tables to add categories or functions in the future, include blank modules in your initial table generation. These modules can later be modified with SECTL to accommodate new categories or functions. See the SECURITY.TBL source module, in Appendix D, for examples of reserving space for extra categories (HP000 and HP001), functions (FMP), and features (SECTL).

3. Put SECURITY into the system generation answer file.

   SECURITY.REL (or your equivalent) must be relocated in the System Message Block. Relocate the modules SECOS.REL and CHECK.REL (shipped on tape) with the other operating system module partitions. Both SECOS.REL and CHECK.REL are partitionable.

4. Reference the Security/1000 libraries in the Library List in your answer file.

   The Security/1000 library SEC1000.LIB (see Appendix G) must be put at the top of the non-CDS list. The non-CDS list should also have SEC1000.LIB at the end. SEC1000CDS.LIB must be at the top of the CDS list, followed by the rest of the CDS libraries, SEC1000.LIB, and ending with BIGLB.LIB. If you do not put the security libraries in the correct place, your programs do not load correctly and undefined externals occur. This can be overcome by referencing the libraries in your list command files.

5. Generate the system and relink all your programs using the new system snap file.

   Be sure to relink SECTL and STGEN as directed in step 1 with the new system snap file.

6. Initialize and turn on Security/1000. This can be done one of two ways depending on your boot procedure.

   a. If your startup program is CI, add the following line as the first line in your Welcome file:

      SECTL +in:<*snapfile*> +on

   b. If you have a startup program other than CI, you should add the following call to your program:

      call SecInitialize(*snapfile*,1,*error*)

      where:

snapfile	is the current system snap file.
1	signals to turn Security/1000 on.
error	is an error code returned by the routine.

7. Boot the new system.

Initializing and Turning On Security/1000

Security/1000 should only be initialized and turned on once at system boot time. This should be done in the welcome file by adding the following:

SECTL +in:<*snapfile*> +on

or programmatically, by using subroutine SecInitialize with the ON option before any users are allowed on the system.

Caution The SECTL IN command, SECTL +IN runstring option, and SecInitialize subroutine should be used with caution. If they are used with a snap file different from the current system snap file, they can corrupt your system and security tables.

The SECTL SW command, SECTL +ON/+OFF runstring options, and SecSwitch subroutine were designed for system engineer debugging and should not be used to turn security on and off after users have been allowed on the system. (System security cannot be guaranteed.)

If you turn on security at some time other than the Welcome file, you may encounter problems. Figure 4-6, Figure 4-7, and the remainder of this section discuss situations of which you should be aware.

In a system with security off, the $CPLV word of the ID segment extension is ignored. That is, there is no check on the PROGCPLV, RQUSCPLV, or ORGCPLV when users wish to run a particular program such as PROG1.

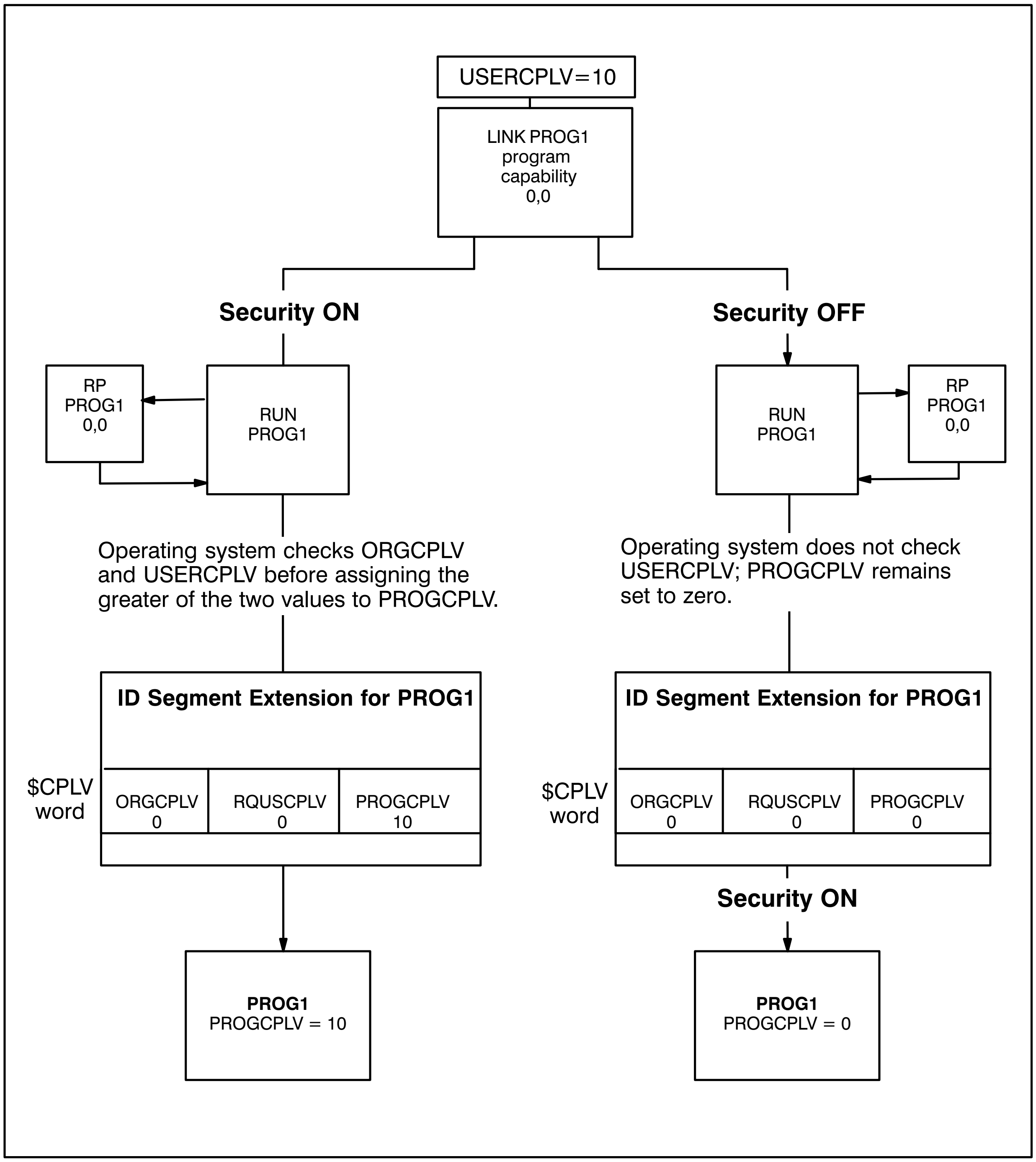
USERCPLV=10
LINK PROG1 program capability 0,0
Security ON
Security OFF
RP PROG1 0,0
RUN PROG1
RUN PROG1
RP PROG1 0,0
Operating system checks ORGCPLV and USERCPLV before assigning the greater of the two values to PROGCPLV.
Operating system does not check USERCPLV; PROGCPLV remains set to zero.
ID Segment Extension for PROG1
$CPLV word
ORGCPLV 0
RQUSCPLV 0
PROGCPLV 10
ID Segment Extension for PROG1
$CPLV word
ORGCPLV 0
RQUSCPLV 0
PROGCPLV 0
Security ON
PROG1 PROGCPLV = 10
PROG1 PROGCPLV = 0

Figure 4-6. Turning On Security/1000

When security is on, the CPLVs in word 47 are checked against the USERCPLV of the user session in which the request was made to determine whether or not a scheduling program may run the child program.

In the example shown in Figure 4-6, the user has a CPLV of 10. When it was linked, PROG1 assumed the PROGCPLV, RQUSCPLV, and ORGCPLV defaults of zero because the user specified no other CPLVs. Because PROGCPLV is set to the higher value of the program's original PROGCPLV and USERCPLV, PROGCPLV is set to 10 when the user tries to run PROG1. USERCPLV must be equal to or greater than RQUSCPLV, as it is in the example. So this user is allowed to run PROG1. PROG1, however, may not be able to do much because of its low CPLV of 10. The user is limited to the capabilities provided in that environment, which may be your intention.

A way to visualize what can happen when security is turned on at a time other than at bootup is to imagine the DS subsystem linked and scheduled with a CPLV of zero before security is turned on. Several users are using DS lines at their terminals. Their programs were linked with default CPLVs of zero because no other CPLVs were specified. Because the programs were scheduled before security was turned on, their PROGCPLVs were not set to the maximum of PROGCPLV and USERCPLV and still have PROGCPLVs of zero. Security is turned on at this point. The DS program lacks the capability to function, programs abort, and users cannot do anything.

The point to remember is that programs in use before Security/1000 is turned on will probably not be able to run if Security/1000 is turned on and they were not linked with a high enough capability level. Of course, if they were linked with a CPLV of 31, there would not be the same problem.

You should be aware of another situation that can occur with programs linked before security is turned on. Users can link programs with any RQUSCPLV and any PROGCPLV when Security/1000 is off. As shown in Figure 4-7, users with a USERCPLV of 20 are linking a program, PROG1, with a RQUSCPLV of 10 and a PROGCPLV of 31.

If security is on when these users link PROG1, they receive the message that they have insufficient capability to use the PC command. PROG1 gets an ORGCPLV and PROGCPLV of zero and a RQUSCPLV of 10.

If security is off when these users link PROG1, security checks are not performed and PROG1 gets an ORGCPLV and PROGCPLV of 31 and a RQUSCPLV of 10. When security is turned on later, users running these programs have access to functions they should not. Users with USERCPLV of 10 or more can use PROG1 with its PROGCPLV of 31. If they have access to the LINK PC command when security is off, you may want to purge users' individual programs before turning on security and have the users relink them after security is on.

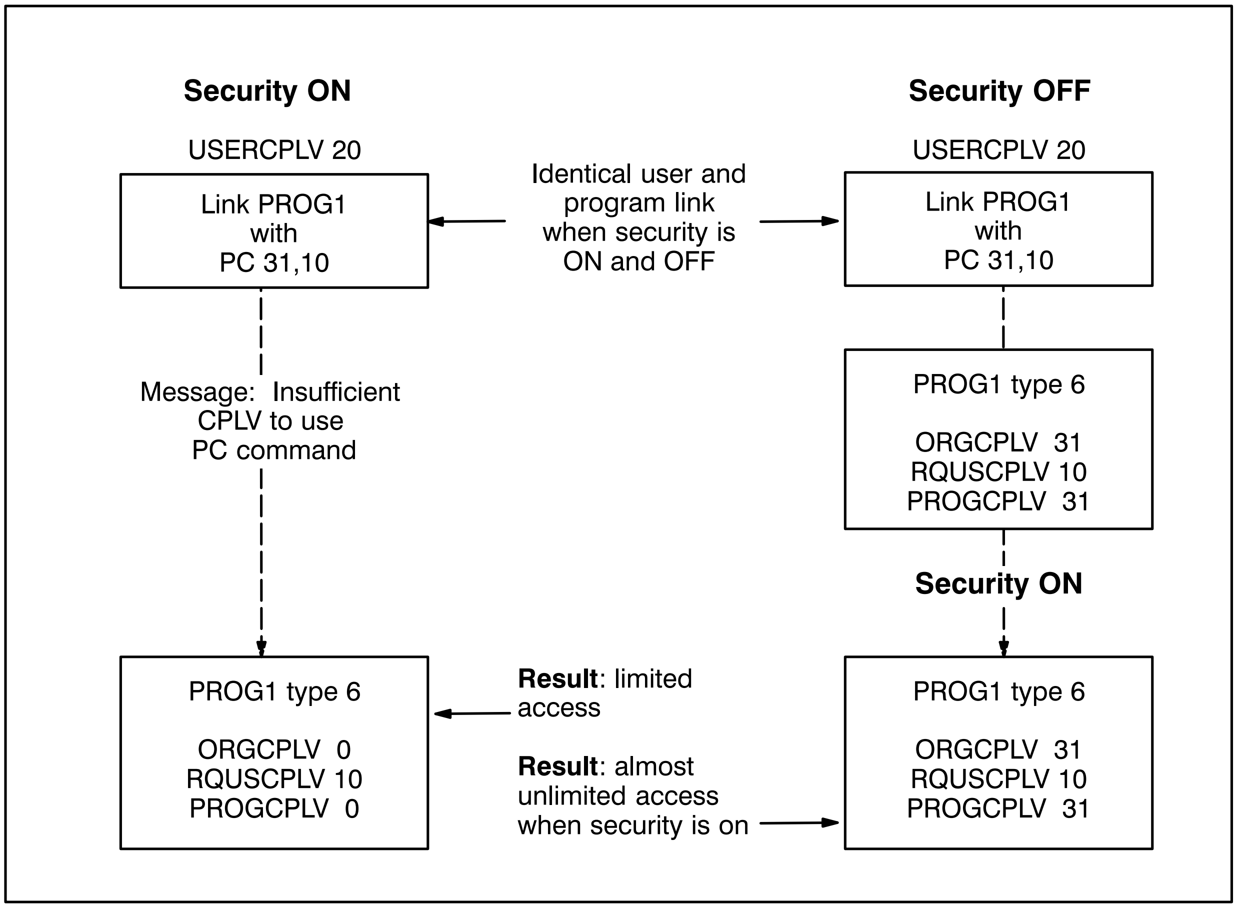

Figure 4-7. Linking Programs and Turning On Security/1000

5

SECTL and STGEN Utilities

SECTL Utility

The SECTL utility performs all the control functions of Security/1000:

- Initialize Security/1000.
- Turn Security/1000 on or off.
- Set or modify the program capability levels.
- Generate a set of security tables.
- List the security tables to a device or file.
- Edit the security tables.
- Display the Security/1000 help file.

Commands issued to SECTL are subject to security checking and only authorized users (those with enough capability) can control Security/1000 operation.

Changes which are made with SECTL are not permanent. When you reboot, the values return to what they were when the system was generated. You must modify the source module for the security table, generate a new system, and reboot to make your changes permanent.

Running SECTL

The SECTL runstring has the following three forms:

Syntax:

```
[RU] SECTL [infile [outfile]]
[RU] SECTL [+IN[:snapfile]] [+ON]
[RU] SECTL +OFF
```

infile	File from which SECTL reads its commands; defaults to the terminal. If it is a disk file, it must have a “SEC” extension.
outfile	File to which SECTL sends its output; defaults to the terminal.

`+IN`	Causes SECTL to initialize Security/1000. (Initialize only once!)
snapfile	Name of current system snapshot file; default is /SYSTEM/SNAP.SNP. (Any other snapshot file can corrupt your system and security tables!)
`+ON`	Turns on Security/1000; can be used only if Security/1000 has been initialized. (Initialize only once!)
`+OFF`	Turns off Security/1000; can be used only if Security/1000 has been initialized. (Only used for system engineer debugging!)

Description:

If the plus options are used in the runstring, SECTL terminates as soon as they have been processed.

SECTL Command Summary

SECTL commands are summarized in Table 5-1.

Table 5-1. SECTL Commands

Command	Purpose	Page #
EC	Modify category function capability	5-3
EX	Terminate SECTL	5-3
GT	Generate security tables	5-4
HE	Display Security/1000 help file	5-4
IN	Initialize Security/1000	5-5
LT	List Security tables	5-5
PC	Set program capability	5-6
RN C	Rename category ID	5-6
RN F	Rename function ID	5-7
RQ	Set required user capability	5-7
SW	Turn on/off Security/1000	5-8

SECTL Commands

Edit Capability Level (EC)

Purpose:	Changes the CPLVs for a given category function.
Syntax:	`EC` *cat_name fnc_name base_cplv* [*sub_cplv1* [*sub_cplv2* [*sub_cplv3*]]]

cat_name	Category defined in currently installed CIT of the security tables.
fnc_name	Function defined in CFT belonging to the specified category.
base_cplv	Base function capability level.
sub_cplv1	Subfunction 1 capability level.
sub_cplv2	Subfunction 2 capability level.
sub_cplv3	Subfunction 3 capability level.

Description:

An example of an EC command is:

```
   SEC > ec grump algrln 20 25
```

In this example, the ALGRLN function in the GRUMP category alters the group logon name. The base function, which defines the CPLV required to change your own group logon name, has been changed to 20. Subfunction 1, which defines the CPLV required to change the group logon name for someone else, has been changed to 25. The values for the second and third subfunction are not changed.

If you want to change the CPLV for the third subfunction and leave the other CPLVs as they are, you can use commas as place holders. The other CPLVs default to their current value. An example of this is:

```
   SEC > ec link pr , , , 25
```

The CPLV of the base function and the first and second subfunctions have not been changed. The CPLV of the third subfunction has been changed to 25.

Exit (EX)

Purpose:	Terminates SECTL.
Syntax:	`EX`

Generate Table (GT)

Purpose: Generates a set of security tables.

Syntax: `GT` *source listfile relfile* [*keepmac*]

source	The name of the file containing the source definition of the security tables.
listfile	The name of the list file. *source*.LST will be the name of the list file if a dash (−) is used. Use zero if you do not want a listing.
relfile	The name of the file that will contain the compiled security tables. *source*.REL will be the name of the relocatable file if a dash (−) is used. Use zero if you do not want a relocatable file.
keepmac	The name of the file where the Macro/1000 code that is generated will be kept. *source*.MAC will be the name of the macro file if a dash (−) is used. Use zero if you do not want a keep file.

Description:

Some examples of the GT command are:

```
SEC > gt security.tbl – – –
```

This example accepts default SECURITY.LST, SECURITY.REL and SECURITY.MAC files. Dashes must be used as place holders.

```
SEC > gt security.tbl tom.lst dick.rel
```

This example sends the listing to TOM.LST, the generated tables to DICK.REL, and does not create a Macro file.

```
SEC > gt security.tbl 0 –
```

This example does not create a listfile, creates the default relfile, and does not create a Macro file.

Help (HE or ?)

Purpose: Displays the Security/1000 help file.

Syntax:
```
HE[LP]
?[?]
```

Initialize (IN)

Purpose:	Initializes Security/1000 using the specified snap file.
Syntax:	`IN [`*snapfile*`] [ON]`

snapfile	File used to initialize Security/1000. If not specified, the default is /SYSTEM/SNAP.SNP.
`ON`	Turns on Security/1000 after initialization. If not specified, Security/1000 is initialized but not turned on.

Description:

Security/1000 must be initialized before it can be turned on. It should only be initialized once at system boot time before any users are allowed on the system.

An example of the IN command is:

```
SEC > in /system/snap.snp on
```

Caution If a snapfile different from the current system snapfile is specified, it can corrupt your system and security tables.

To initialize and turn on the security each time the system is booted, the following line *mus*t be the first command in the Welcome file:

```
RU SECTL +IN :<snap file name> +ON
```

List Table (LT)

Purpose:	Lists the currently installed security tables.
Syntax:	`LT [`*listfile*`]`

listfile	Destination file for the list of security tables. If the file exists, it is overwritten. If the file does not exist, it is created. If not specified, the default is LU 1.

Examples of the LT command are:

```
SEC > lt tom.lst

SEC > lt jamala.lst
```

Program Capability (PC)

Purpose: Sets the program capability level.

Syntax: `PC` *program new_progcplv*

program	Program whose capability level is to be reset in the ID segment extension.
new_progcplv	New capability level.

Description:

The value of the PROGCPLV cannot exceed that of the USERCPLV for the person running SECTL. The program must have an ID segment that will be updated. The program file is not updated with the PC command. You must use LINK to update the program file.

An example of the PC command is:

```
   SEC > pc setup 31
```

In this example, the PC command sets the PROGCPLV for the SETUP program to 31. Since the PROGCPLV defined cannot be greater than the USERCPLV of the person using SECTL, the user in this example must have a USERCPLV of 31.

Rename a Category (RN C)

Purpose: Renames a category in the category index table (CIT).

Syntax: `RN C` *old_cat_name new_cat_name*

old_cat_name	Name of the category to be renamed in the CIT.
new_cat_name	New category name.

Description:

The old category name must exist in the CIT and the new category name must not.

In the following example, category CAT1 in the CIT is renamed to CATA:

```
   SEC > rn c cat1 catA
```

Do not rename standard categories such as CI, or all security becomes ineffective.

Rename a Function (RN F)

Purpose:	Renames a function in a Category Function Table (CFT).
Syntax:	`RN F` *cat_name old_fnc_name new_fnc_name*

cat_name	Name of the category containing the function to be renamed.
old_fnc_name	Function in the CFT to be renamed.
new_fnc_name	New function name.

Description:

The old function name must exist in the CFT and the new function name must not.

In the following example, the CMD1 function in the CAT1 category is renamed to CMDA:

```
SEC > rn f cat1 cmd1 cmda
```

You should not rename standard functions, or all security becomes ineffective.

Required User Capability (RQ)

Purpose:	Sets the program's required user capability level (RQUSCPLV).
Syntax:	`RQ` *program new_rquscplv*

program	Program whose RQUSCPLV is to be set.
new_rquscplv	New RQUSCPLV.

Description:

The value of RQUSCPLV cannot exceed the USERCPLV of the user running SECTL. The program must have an ID segment that will be updated. The program file is not updated with the RQ command. You must use LINK if you want to update the program file.

In the example,

```
SEC > rq setup 28
```

the RQUSCPLV of the SETUP program is set to 28. Because the RQUSCPLV of a program cannot be greater than the USERCPLV of the person running SECTL, the user must have a USERCPLV of at least 28.

Switch (SW)

Purpose: Switches Security/1000 on or off.

Syntax: `SW ON|OFF`

`ON`	Switches on Security/1000.
`OFF`	Switches off Security/1000.

Description:

This command can be used only after Security/1000 has been initialized. (Security should be initialized and turned on at system boot up before any users are allowed on the system.) Use of this command is governed by the required CPLV defined for the SW command in the SECTL category whether security is on or off.

Caution This command was designed for system engineer debugging and should not be used to turn security on or off after users have been allowed on the system. System security cannot be guaranteed if this occurs.

Asterisk (*)

Purpose: An * in column one means the line is a comment. This is used after SECTL commands are placed in a command file.

Description:

An example of the use of the asterisk (*) is as follows:

```
* The following commands create
* category CRDGP from extra category
* EXTRA
  rn c EXTRA crdgp
  rn f crdgp res00 nogroup
  rn f crdgp res01 system
  ec crdgp system 0 1 2 3
  ex
* End of creation.
*
```

The above example assumes category EXTRA, with functions res00 and res01, already exists in the system security tables.

STGEN Utility

When installing Security/1000, you can generate security tables directly by running the STGEN program rather than using the SECTL GT command.

STGEN can be used if you want to generate tables from a CI command file and want to know if there are any errors and what those errors are. STGEN returns all error codes to CI in the $RETURN1 through $RETURN5 variables.

The STGEN runstring is as follows:

Syntax: `[RU] STGEN` *source listfile relfile keepmac*

source	The name of the file containing the source definition of the security tables.
listfile	The name of the list file. *source*.LST is the list file if a dash (–) is specified. Use zero if you do not want a listing.
relfile	The name of the file that will contain the compiled security tables. *source*.REL will be the name of the relocatable file if a dash (–) is used. Use zero if you do not want a relocatable file.
keepmac	The name of the file where the Macro/1000 code that is generated will be kept. *source*.MAC will be the name of the macro file if a dash (–) is used. Use zero if you do not want a keep file.

Example of Security Table Source

This is a sample of the source definition of a security table.

An asterisk in the first column means the text in that line is a comment. Blank lines are allowed.

```
*-----<  The security category for GRUMP.
*
*
*-----<  Meanings of the base and subfunctions are:
*-----<
*-----<  (1) Base  - Capability needed to perform function on self.
*-----<      Sub01 - Capability needed to perform function on
*-----<              someone else
*-----<      Sub02 - Not defined
*-----<      Sub03 - Not defined
*-----<
*-----<  (2) Base  - Capability needed to perform function on a group
*-----<              in which you are a member
*-----<      Sub01 - Capability needed to perform function on a group
*-----<              in which you are not a member
*-----<      Sub02 - Not defined
*-----<      Sub03 - Not defined
$category: grump

   algrln  31  31  0   0   *alter group logon name               (2)
   algrcp  25  31  0   0   *alter group cpu limit                (2)
   algrco  25  31  0   0   *alter group connect time limit       (2)
   algrbm  25  31  0   0   *alter group bit map                  (2)
   alusln  31  31  0   0   *alter logon user name                (1)
   alusrn  10  31  0   0   *alter logon user real name           (1)
   aluspw  10  31  0   0   *alter user password                  (1)
   alusud  25  31  0   0   *alter user udsp depth and levels     (1)
   aluscl  31  31  0   0   *alter user cplv                      (1)
   alusbm  25  31  0   0   *alter user bit map                   (1)
   aluscp  31  31  0   0   *alter user cpu limit                 (1)
   alusco  31  31  0   0   *alter user connect limit             (1)
   alusgr  25  31  0   0   *alter user group list (ie add groups) (1)
   aludlg  15  31  0   0   *alter user default logon group       (1)
   alussu  15  31  0   0   *alter user startup program           (1)
   aluslf  15  31  0   0   *alter user logoff program            (1)
   aluswd  31  31  0   0   *alter user logon working directory   (1)
   ligrp   15  31  0   0   *list group definitions               (2)
   liusr   15  31  0   0   *list user definitions                (1)
   passwd   0  25  0   0   *change user password                 (1)*

*-----<  The meanings of the base and subfunctions are now:
*-----<
*-----<  Base  - Capability needed to issue the command.
*-----<  Sub01 - Not defined.
*-----<  Sub02 - Not defined.
*-----<  Sub03 - Not defined.
*
   negrp   31  0   0   0   *create new group
   neusr   31  0   0   0   *create new user
   pugrp   31  0   0   0   *purge a group
   puusr   31  0   0   0   *purge a user
   regrp   31  0   0   0   *reset a groups accounting info
   reusr   31  0   0   0   *reset a users accounting info
   kilses  31  0   0   0   *kill a session
*
```

Example of Generated Macro/1000 Code

This is the Macro/1000 code generated by STGEN or the SECTL GT command for the previous sample of security table source code.

```
macro,l,c
                 hed SECURITY/1000 Table

*
*-----<Source file:         SECURITY.TBL
*-----<List file:           SECURITY.lst
*-----<Relocatable file:    SECURITY.rel:::5
*-----<keep file:           SECURITY.mac
*

                 nam sctbl 0  92078-1X102  Rev.5000 861016.225844
                 ent CatIndex $sysct.adr $exect.adr
                 ent $secct.adr $fmpct.adr
GRUMP.adr        dec 24
                 asc 3 ALGRLN
                 dec 31
                 dec 31
                 dec 0
                 dec 0
                 asc 3 ALGRCP
                 dec 25
                 dec 31
                 dec 0
                 dec 0
                 asc 3 ALGRCO
                 dec 25
                 dec 31
                 dec 0
                 dec 0
                 asc 3 ALGRBM
                 dec 25
                 dec 31
                 dec 0
                 dec 0
                 asc 3 ALUSLN
                 dec 31
                 dec 31
                 dec 0
                 dec 0
```

```
asc 3 ALUSRN
dec 10
dec 31
dec 0
dec 0
asc 3 ALUSPW
dec 10
dec 31
dec 0
dec 0
asc 3 ALUSUD
dec 25
dec 31
dec 0
dec 0
asc 3 ALUSCL
dec 31
dec 31
dec 0
dec 0
asc 3 ALUSBM
dec 25
dec 31
dec 0
dec 0
asc 3 ALUSCP
dec 31
dec 31
dec 0
dec 0
asc 3 ALUSCO
dec 31
dec 31
dec 0
dec 0
asc 3 ALUSSU
dec 15
dec 31
dec 0
dec 0
asc 3 ALUSLF
dec 15
dec 31
dec 0
dec 0
asc 3 ALUSWD
dec 31
dec 31
dec 0
dec 0
asc 3 LIGRP
dec 15
dec 31
dec 0
dec 0
asc 3 LIUSR
dec 15
dec 31
dec 0
dec 0
asc 3 NEGRP
dec 31
dec 0
dec 0
dec 0
asc 3 NEUSR
dec 31
dec 0
dec 0
dec 0
asc 3 PUGRP
dec 31
dec 0
dec 0
dec 0
asc 3 PUUSR
dec 31
dec 0
dec 0
dec 0
asc 3 REGRP
dec 31
dec 0
dec 0
dec 0
asc 3 REUSR
dec 31
dec 0
dec 0
dec 0
asc 3 KILSES
dec 31
dec 0
dec 0
dec 0
```

A

Modifying Security/1000 Answer Files

To install Security/1000, you must modify your answer file. Add the information shown below in boxes to your current answer file:

```
.
.
.
pa  cdsfh,time,class,spslg,alarm ,SECOS,CHECK

ms $sysa:92077
end
*
* OS partitions
*
.
.
.
```

```
re,SECOS.rel
en
*
re,CHECK.rel
en
```

```
.
.
.
*******************
* System Messages *
*******************
re,%msgtb
en
re,%$m000
en
```

```
re,security.rel
en
```

```
.
.
.
en
```

```
*******************
* System Libraries *
*******************

lib bigds.lib
```

```
lib SEC1000.lib
```

```
lib fndlb.lib
lib biglb.lib

end
***********************
* CDS System Libraries *
***********************
```

```
lib SEC1000CDS.lib
```

```
lib bgcds.lib
```

```
lib SEC1000.lib
```

```
lib bigds.lib
lib fndlb.lib
lib biglb.lib
*
* end
```

B

Logon Files

```
**********************************************************************
**                                                                  **
** LOGON File for James Alabama                                     **
**                                                                  **
** Last changed <870730.1752>                                       **
**                                                                  **
**********************************************************************
*
* First set the logging to the screen to off
SET LOG = OF
* Logoff user after 12 timeouts (3 x 4)
SET AUTO_LOGOFF = 3
*
* Set a variable called ’user_name’ that can be used to identify user
*
SET USER_NAME = JAMES
*
* Make sure that the stack is saved
*
SET SAVE_STACK = TRUE
*
* Set the prompt to be the user’s name
*
SET PROMPT = $USER_NAME>
*
* Set UDSP first to working directory, then to system, and then to
* user dir.
*
PATH 1 .,/PROGRAMS,/$USER_NAME/PROGRAMS
PATH 2 .,/CMDFILE,/$USER_NAME/CMDFILES
*
* Echo a friendly message
ECHO ’Hello ’ $USER_NAME
RETURN
```

Global Logon File

```
**********************************************************************
**                                                                  **
** GLOBAL_LOGON file for all users                                  **
**                                                                  **
** Last changed <870730.1753>                                       **
**                                                                  **
**********************************************************************
*
* A message that LUs are being verified or backed up may be written
* here and then an ’EX’ can be put in to keep all users off the system.
*
*
* Show the users the system messages.
LI /SYSTEM/SYSTEM_MESSAGE
*
* Other messages or commands can be executed for all users to see
*
* Note that the next command looks into the user’s default working
* directory for the personal logon file.
IF DL LOGON.CMD,,0
   THEN TR,LOGON.CMD
FI
*
*
*
```

C

GRUMP Command/Log Files

Example of GRUMP Command File

```
*  This is an example of a GRUMP command file.
*  It creates group MAX.
*  Then user DEDRA is created and added to group MAX during
*    the creation of her user account.
*  Group CPE is created.
*  Then DEDRA is added to group CPE by altering her existing
*    user account.
*  DEDRA’s user account is given the password ”bluejay”
*    during creation and that password is changed to no
*    password during the alteration of her user account.
*  The user and group accounts created are then purged from
*    the multiuser system.
*
*  All lines beginning with an asterisk are comments
*    included to explain what is happening.
*
*  Assumptions made for this file to work:
*       1. User DEDRA does not exist.
*       2. Groups MAX and CPE do not exist.
*       3. Directories /DEDRA and /DEDRA/MAX do not exist.
*       4. The command file name is TREX.
*
*  Usage:   RU,GRUMP,TREX
*
*  If you want to see the corresponding log file, the usage
*       string would be:
*
*              RU,GRUMP,TREX,LOGTREX
*
*  Make sure that file LOGTREX does not exist before
*       specifying it as the log file.
*
*
* >> Create group MAX and provide group information.
*
NEW,GROUP,MAX,4:30:0,500:0,–100:255,/e
*
* >> Create user DEDRA.
*
NEW,USER,DEDRA
*
```

```
* >> Provide unique user information.
*
‘Dedra Jackson‘,BLUEJAY,25,–200:250,220,225,/E,4:4,2
*
* >> Provide DEDRA.NOGROUP information.  Directory ::DEDRA
* >> does not exist and we want to create it on LU 17 (which
* >> explains the ”YES,17” response).
*
* >> If directory ::DEDRA already exists, the input line
* >> must be
* >> ”::DEDRA,‘RU CI.RUN::PROGRAMS LOGON.CMD::USERS‘”.
*
* >> If directory ::DEDRA does not exist and you do not want
* >> to create it, the beginning of the input line must be
* >> ”::DEDRA,NO,‘RU CI.RUN::PROGRAMS LOGON.CMD::USERS‘”.
*
* >> Note that ::DEDRA is the default for the working directory
* >> so ”,YES,17,‘RU CI.RUN::PROGRAMS LOGON.CMD::USERS’”
* >> gives the same result as the line below.
*
* >> If a command extends beyond one line, GRUMP will parse
* >> a parameter–separating comma at the end of a line as an
* >> undefined parameter and result in error.  Omit end of
* >> line commas if they are separators rather than place
* >> holders.  Continue the command in column one of the next
* >> line.
*
::DEDRA,YES,17,‘RU CI.RUN::PROGRAMS LOGON.CMD::USERS‘
‘LOGOF.CMD::DEDRA‘,0:20:0,50:0
*
* >> YES we want to add DEDRA to an existing group other
* >> than NOGROUP.
*
YES
*
* >> Provide the group name, MAX, and the DEDRA.MAX information.
*
MAX,/DEDRA/MAX,YES,,‘RU CI.RUN::PROGRAMS LOGON.CMD‘,,–1,–1
*
* >> YES MAX should be the default logon group.
*
YES
* >> /E says we do not want to add DEDRA to any other
* >> existing groups.
*
/E
*
* >> Create group CPE and provide group information.
*
NEW,GROUP,CPE,0:45:0,400:0,–100:255,/E
*
* >> Alter unique user information for DEDRA so we can add
* >> her to the existing group CPE.
*
* >> Leave all the unique user attributes the same except
* >> change the password which was previously defined, to
* >> no password (this explains the YES).
*
ALTER,USER,DEDRA.,,,,YES,,,,,
*
```

```
* >> YES we want to add DEDRA to an existing group.
*
YES
*
* >> Provide group name and DEDRA.CPE information.  Directory
* >> ::DEDRA already exists at this point so GRUMP does not
* >> ask whether we want to create it and which LU it should
* >> go on.
*
CPE,::DEDRA,'RU CI.RUN::PROGRAMS LOGON.CMD::DEDRA','LOGOF.CMD::CPE'
–1,–1
*
* >> No we do not want to add DEDRA to any other existing
* >> groups (this explains the /E).
*
/E
*
* >> CPE should be the default logon group
*
CPE
*
* >> Purge user DEDRA
*
PU,USER,DEDRA,OK
*
* >> Purge group MAX
*
PU,GROUP,MAX,OK
*
* >> Purge group CPE
*
PU,GROUP,CPE,OK
*
* >> Exit GRUMP
*
EX
```

Example of Log File

This is an example of a log file. This log file would be created if specified in the RUN command with GRUMP command file immediately preceding.

```
* GRUMP>
NEW
* Enter (G)roup or (U)sers :
GROUP
* Enter group logon name :
MAX
* Creating group MAX.
* Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit]:
4:30:0
* Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit]:
500:0
* LU access table modifications ( [–]LU#[:LU#2] ) :
–100:255
* LU access table modifications ( [–]LU#[:LU#2] ) :
```

```
/E
* GRUMP>
NEW
* Enter (G)roup or (U)sers :
USER
* Enter user logon name :
DEDRA
* Creating user DEDRA
* Enter users real name [???] :
Dedra Jackson
* Enter password (a <cr> gives no password) :
BLUEJAY
* Enter capability level (31=SU) [10] :
25
* LU access table modifications ( [–]LU#[:LU#2] ) :
–200:250
* LU access table modifications ( [–]LU#[:LU#2] ) :
220
* LU access table modifications ( [–]LU#[:LU#2] ) :
225
* LU access table modifications ( [–]LU#[:LU#2] ) :
/E
* Enter #UDSPs:depth [0:0] :
4:4
* Enter the size of the Environment Variable Block in pages [0]:
2
*
* All users must be in the group NOGROUP.
*
*
* Enter information for DEDRA.NOGROUP
*
* Enter working directory name [::DEDRA] :
::DEDRA
* Create directory ::DEDRA (Yes/No) [N] :
YES
* What LU should the directory go on [0] :
17
* Enter the startup command [RU CI.RUN::PROGRAMS] :
RU CI.RUN::PROGRAMS LOGON.CMD::USERS
* Enter the logoff program/command file [NOT DEFINED] :
LOGOF.CMD::DEDRA
* Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit] :
0:20:0
* Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit] :
50:0
*
* Do you wish to include the user in any existing
* group other than NOGROUP (Yes/No) [N] :
YES
*
* Enter group name (/E or <cr> to end) :
MAX
*
* Enter information for DEDRA.MAX.
*
* Enter working directory name [::DEDRA] :
/DEDRA/MAX
* Create directory (Yes/No) [N] :
YES
```

```
* What LU should the directory go on [0] :

* Enter the startup command [RU CI.RUN::PROGRAMS] :
RU CI.RUN::PROGRAMS LOGON.CMD::USERS
* Enter the logoff program/command file [NOT DEFINED] :

* Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit] :
–1
* Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit] :
–1
* Should this be the default logon group (Yes/No) [N] :
YES
*
* Enter group name (/E or <cr> to end) :
/E
* Group MAX is the default logon group.
* GRUMP>
NEW
* Enter (G)roup or (U)sers :
GROUP
* Enter group logon name :
CPE
* Creating Group CPE.
* Enter CPU limit (hh:mm:ss or –1 for No Limit) [No Limit] :
0:45:0
* Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit] :
400:0
* LU access table modifications ( [–]LU#[:LU#2] ) :
–100:255
* LU access table modifications ( [–]LU#[:LU#2] ) :
/E
* GRUMP>
ALTER
* Enter (G)roup or (U)sers :
USER
* Enter user.group parameter:
DEDRA.
* Enter new user logon name [DEDRA] :

* Enter users real name [Dedra Jackson] :

* Enter password (a <cr> gives no password) :

* Change password to no password (Yes/No) [N] :
YES
* Enter capability level (31=SU) [25] :

* LU access table modifications ( [–]LU#[:LU#2] ) :

* Enter #UDSPs:depth [4:4] :

* Enter the size of the Environment Variable Block in pages [2]:

* Modified unique user information.
*
* Do you wish to include the user in any existing
* group other than NOGROUP (Yes/No) [N] :
YES
* Enter group name (/E or <cr> to end) :
CPE
```

```
*
* Enter information for DEDRA.CPE.
*
* Enter working directory name [::DEDRA] :
::DEDRA
* Enter the startup command [RU CI.RUN::PROGRAMS] :
RU CI.RUN::PROGRAMS LOGON.CMD::DEDRA
* Enter the logoff program/command file [NOT DEFINED] :
LOGOF.CMD::CPE
* Enter CPU limit (hh:mm:ss or –1 for No Limit) [] :
–1
* Enter connect time limit (hh:mm:ss or –1 for No Limit) [No Limit] :
–1
*
* Enter group name (/E or <cr> to end) :
/E
*
* Which group should be the default logon group [MAX] :
CPE
* Group CPE is the default logon group.
* GRUMP>
PU
 *Enter (G)roup or (U)sers :
USER
* Enter user[.[group][,OK]] :
DEDRA
OK
* DEDRA.NOGROUP record purged
* DEDRA.MAX record purged
* DEDRA.CPE record purged
* User DEDRA successfully purged.
* GRUMP>
PU
* Enter (G)roup or (U)sers :
GROUP
* Enter group[,OK] :
MAX
OK
* Purged group MAX
* GRUMP>
PU
* Enter (G)roup or (U)sers :
GROUP
* Enter group[,OK] :
CPE
OK
* Purged group CPE.
* GRUMP>
EX
```

D

SECURITY.TBL

SECURITY.TBL is the security source code shipped with the software. It is used with the SECTL GT command to generate a set of tables at installation.

Security Table Source Format

The general format is free form, with exceptions only where noted. Input is NOT case sensitive; it is all upshifted internally.

The NAME fields are all up to six bytes long. They may consist of any ASCII characters except carriage return and line feed.

Any line with an asterisk (*) in column one is considered a comment. Comments may appear at the end of any source line, provided that ALL fields of the source line were specified.

Categories

Categories are defined with the $CATEGORY keyword. The format is:

```
   $category: <category name> [rangeoff]
```

$category can start anywhere on the line. There must be at least one space between the colon (:) and the start of the category name. The rangeoff option instructs the table generator to allow values outside the 0 to 31 range. This option is used when the functions in the category are used to hold information other than CPLVs. Rangeoff cannot be used on special categories.

Special Categories

There are four special categories, each indicated by a keyword. Note that the colon (:) is part of the keyword. The rangeoff option cannot be specified with the following categories:

```
   $system:     OS commands and signal security
   $exec:       Exec call security
   $fmp:        File system security
   $security:   Security/1000 internal security checking
```

Special categories can appear in any order in the source file and the functions within the category can appear in any order. However, the table generator reorders the special categories and the functions within them to an order that is known to the system software (for performance reasons).

Only certain functions (predefined) can appear within a special category. These are known to the table generator. If any other function appears in a special category it generates an error condition. The predefined functions for each special category are listed below.

Some of the special categories have reserved functions in them. Currently, no software makes use of them. They exist for possible future expansion of the functions in the special categories. The reserved functions appear in the list file produced by SECTL LT command or the SecLisTable subroutine.

`$system:` Each function corresponds to an OS command except sigkil, which is used by signals to determine who can send a kill signal.

```
br, cd, dt, go, of, pr, ss, sz, ul,
up, vs, ws, as, dn, ds, ps, tm, ru,
xq, ex, sigkil
```

`$exec:` Each function corresponds to an EXEC code. Note that even if an EXEC code is currently undefined, it is still given an entry within this category. Note also that $exec is currently an inactive category, which means that it is not presently used by HP software.

```
exec01, exec02, exec03, ......
......, exec44
```

`$fmp:` Each function corresponds to an FMP function. See the following SECURITY.TBL section for the definitions of the FMP functions.

```
fmp00, fmp01, fmp02, ......
....., fmp35
```

`$security:` Because the functions in this category represent a variety of things, each function and its meaning is listed:

Function	Meaning
edtfnc	subroutine SecEditFunction
rncat	subroutine SecRenameCat
rnfnc	subroutine SecRenameFnc
putprg	subroutine SecPutProgCplv
putrq	subroutine SecPutRqusCplv
putcit	subroutines SecPutCitNam, SecPutCitNum
putcft	subroutines SecPutCftNam, SecPutCftNum
ckcplv	subroutines SecChkCplvNam, SecChkCplvNum
inital	subroutine SecInitialize
switch	subroutine SecSwitch
vfnam	subroutine VfNam
clgon	subroutine Clgon
clgof	subroutine Clgof

Functions

Each category contains a series of functions. There must be at least one function per category. The functions follow the category definition. All functions are linked to the last seen category definition. The syntax of the function statement is as follows:

fncname <base> sub1 sub2 sub3

fncname	The name of the function; must be unique within the current category.
base	The cplv of the base function; must be in the range 0-31, unless rangeoff was specified.
sub1	The cplv of subfunction1; must be in the range 0-31, unless rangeoff was specified.
sub2	The cplv of subfunction2; must be in the range 0-31, unless rangeoff was specified.
sub3	The cplv of subfunction3; must be in the range 0-31, unless rangeoff was specified.

For example,

```
putcit   12   1   2   0
```

SECURITY.TBL

This is a sample of the security source code shipped with the software. Use it to generate a set of tables at installation.

```
*
*-----< Security definition for programmatic access to the file
*-----< system via the FMP library.
*-----< The following meanings have been applied to the base function
*-----< and or subfunctions.
*-----<
*-----<     (1)    Base – caller can either call the routine or not.
*-----<            Sub1 – action can be performed in the current WD.
*-----<            Sub2 – action can be performed on the same LU as
*-----<                   the current WD.
*-----<            Sub3 – action can be performed on any LU.
*-----<
*-----<     (2)    Base – caller can either call the routine or not.
*-----<            Sub1 – action can be performed by the owner.
*-----<            Sub2 – action can be performed by another member of
*-----<                   the owner’s group.
*-----<            Sub3 – action can be performed by any other user on
*-----<                   the system.
*-----<
*-----<     (3)    Base – caller can either call the routine or not.
*-----<            Sub1 – not defined
*-----<            Sub2 – not defined
*-----<            Sub3 – not defined
*

$fmp:

          create   10 10 11 13    * FmpOpen – create mode          (1)
          open     1  1  2  4     * FmpOpen – open mode            (1)
          fmp02    0  0  0  0     * reserved for future use
          purge    10 10 11 13    * FmpPurge                       (1)
          unprg    10 10 11 13    * FmpUnPurge                     (1)
          init     20 20 25 30    * FmpMount – initialize mode     (2)
          mount    5  0  0  0     * FmpMount – mount only mode     (2)
          dismnt   5  0  0  0     * FmpDismount                    (2)
          crdir    10 10 11 13    * FmpCreateDir                   (1)
          wd       5  0  0  0     * FmpWorkingDir                  (3)
          acctim   5  0  0  0     * FmpAccessTime                  (3)
          updtim   5  0  0  0     * FmpUpdatetime                  (3)
          cretim   5  0  0  0     * FmpCreatetime                  (3)
          setown   10 0  0  0     * FmpSetOwner                    (3)
          eof      5  0  0  0     * FmpEof                         (3)
          size     5  0  0  0     * FmpSize                        (3)
          setwd    15 15 16 18    * FmpSetWorkingDir               (1)
          info     5  0  0  0     * FmpInfo                        (3)
          setdir   10 0  0  0     * FmpSetDirInfo                  (3)
          reccnt   5  0  0  0     * FmpRecordCount                 (3)
          filenm   5  0  0  0     * FmpFileName                    (3)
          fmp21    0  0  0  0     * reserved for future use
          rename   10 10 11 13    * FmpRename                      (1)
          fmp23    0  0  0  0     * reserved for future use
          trunct   10 0  0  0     * FmpTruncate                    (3)
          prot     5  0  0  0     * FmpProtection                  (3)
          setprt   10 0  0  0     * FmpSetProtection               (3)
          opfile   5  0  0  0     * FmpOpenFiles                   (3)
          reclen   5  0  0  0     * FmpRecordLen                   (3)
          fmp29    0  0  0  0     * reserved for future use
          fmp30    0  0  0  0     * reserved for future use
          fmp31    0  0  0  0     * reserved for future use
          dirnam   5  0  0  0     * FmpDirAddToNam                 (3)
          fmp33    0  0  0  0     * reserved for future use
          fmp34    0  0  0  0     * reserved for future use
          fmp35    0  0  0  0     * reserved for future use
          fmp36    0  0  0  0     * reserved for future use
          fmp37    0  0  0  0     * reserved for future use
          fmp38    0  0  0  0     * reserved for future use
          fmp39    0  0  0  0     * reserved for future use
          fmp40    0  0  0  0     * reserved for future use
          fmp41    0  0  0  0     * reserved for future use
          fmp42    0  0  0  0     * reserved for future use
          fmp43    0  0  0  0     * reserved for future use
          fmp44    0  0  0  0     * reserved for future use
          fmp45    0  0  0  0     * reserved for future use
          fmp46    0  0  0  0     * reserved for future use
          fmp47    0  0  0  0     * reserved for future use
          fmp48    0  0  0  0     * reserved for future use
          fmp49    0  0  0  0     * reserved for future use
          fmp50    0  0  0  0     * reserved for future use
          fmp51    0  0  0  0     * reserved for future use
          fmp52    0  0  0  0     * reserved for future use
          fmp53    0  0  0  0     * reserved for future use
          fmp54    0  0  0  0     * reserved for future use
          fmp55    0  0  0  0     * reserved for future use
          fmp56    0  0  0  0     * reserved for future use
          fmp57    0  0  0  0     * reserved for future use
          fmp58    0  0  0  0     * reserved for future use
          fmp59    0  0  0  0     * reserved for future use
          fmp60    0  0  0  0     * reserved for future use
          fmp61    0  0  0  0     * reserved for future use
          fmp62    0  0  0  0     * reserved for future use
          dsrtr    15 0  0  0     * ds/1000 file transparency      (3)

*
*-----< The System commands security category. These are the commands
*-----< implemented in the kernel. The program that issues the OS command
*-----< is called the "sender" and the program that the command will affect
*-----< is called the "receiver".
*-----<
*-----< The following meanings have been applied to the base function
*-----< and or subfunctions. The sender's PROGCPLV is always checked against
*-----< the Base, if that check passes the check is against one of the
*-----< subfunctions. Which subfunction is used depends on the parameters.
*-----<
*-----<   Base  - The sender can either issue the command or not.
*-----<
*-----<   Sub01 - The sender and the receiver are in the same session.
*-----<
*-----<   Sub02 - The sender and the receiver are in different sessions but
*-----<           they are both in the same group.
```

```
*-----<
*-----<   Sub03 – The sender and the receiver are in different sessions and
*-----<           different groups.
*


$system:
          as      15  15  20  25
          br      5   5   20  25
          cd      6   6   11  16
          dt      10  10  20  25
          go      5   5   10  15
          of      10  10  20  25
          pr      20  20  20  25
          ss      10  10  20  25
          sz      6   6   11  16
          ul      20  0   0   0
          up      5   0   0   0
          vs      7   7   20  25
          ws      12  12  20  25
          dn      15  0   0   0
          ds      5   0   0   0
          ps      0   0   0   0
          tm      1   31  0   0
          ru      8   8   0   0   * sub02 and sub03 undefined for this cmd.
          xq      8   8   0   0   * sub02 and sub03 undefined for this cmd.
          ex      0   0   0   0
          sglkil  10 10  20  20   * Signal kill function

*
*-----< The first of two categories for SECURITY/1000
*-----< This category defines who can use the SECURITY/1000 routines and
*-----< what they can do with them.
*

$security:

              edtfnc  31  0   0   0       * SecEditFunction
              rncat   31  0   0   0       * SecRenameCat
              rnfnc   31  0   0   0       * SecRenameFnc


*
*-----< SecPutProgCplv and SecPutRqusCplv have the base function and
*-----< the subfunction defined. Their meanings are as follows.
*-----<
*-----<   Base    – The request applies to the calling program.
*-----<
*-----<   Sub01   – The request applies to another program in the
*-----<             same session as the caller.
*-----<
*-----<   Sub02   – The request applies to a program in another
*-----<             session but the same group as the caller.
*-----<
*-----<   Sub03   – The request applies to a program in another
*-----<             session and group from the caller.
*
```

```
              putprg  5   15  31  0       * SecPutProgCplv
              putrq   5   15  31  0       * SecPutRqusCplv
              putcit  31  0   0   0       * SecPutCitNam and SecPutCitNam
              putcft  31  0   0   0       * SecPutCftNam and SecPutCftNam
              ckcplv  0   0   0   0       * SecChkCplvNam and SecChkCplvNum
              inital  31  0   0   0       * Secinitial
              switch  31  0   0   0       * SecSwitch


*
*-----< VfNam, Clgon, Clgof.
*-----<
*-----<   Base    - Yes/No
*-----<
*-----<   Sub01   - Log directive. If non-zero then any security check that
*-----<             fails will be logged to the spool log file.
*-----<
*-----<   Sub02   - Abort Directive. If 0 nothing will be aborted. If 1 the
*-----<             program that failed the security check will be aborted.
*-----<
*


              vfnam   20  1   1   0       * VfNam
              clgon   20  1   1   0       * Clgon
              clgof   20  1   1   0       * Clgof


*
*-----< The second SECURITY/1000, it is used to determine who can
*-----< use the Sectl commands and what they can do with them.
*

$category: sectl

              in     31  0   0   0   * IN command
              sw     31  0   0   0   * SW command
              ec     31  0   0   0   * EC command
              he     0   0   0   0   * HE command
              gt     5   0   0   0   * GT command
              lt     5   0   0   0   * LT command
              pc     31  0   0   0   * PC command
              rn     31  0   0   0   * Rn command
              rq     31  0   0   0   * RQ command
              res000 0   0   0   0   * Spares for future features
              res001 0   0   0   0   * Spares for future features
              res002 0   0   0   0   * Spares for future features
              res003 0   0   0   0   * Spares for future features
              res004 0   0   0   0   * Spares for future features
              res005 0   0   0   0   * Spares for future features
              res006 0   0   0   0   * Spares for future features
              res007 0   0   0   0   * Spares for future features
              res008 0   0   0   0   * Spares for future features
              res009 0   0   0   0   * Spares for future features
```

```
*
*-----< Link security table
*

$category: link

              os     5   0   0   0   * operator suspend link

*
*-----<  PR command functions have the following meanings.
*-----<
*-----<  Base   –  The user can either issue this command or not. If so,
*-----<            then the user is subject to the subfunction meanings.
*-----<
*-----<  sub01  –  The user can only decrease the default priority
*-----<            ( 99 –> 32k ).
*-----<
*-----<  sub02  –  The user can increase the priority to 51
*-----<
*-----<  sub03  –  The user can set any priority.
*

              pr     15  15  20  28  * set priority of program

              lc     20  0   0   0   * labelled system common
              sc     20  0   0   0   * blank system common
              sh     20  0   0   0   * Use Sharable EMA

*
*-----< CI security tables. All CI commands only use the base function.
*-----< Therefore for each command it is a simple YES/NO decision.
*-----<
*-----< Interdependencies. A number of CI commands map directly onto FMP
*-----< routines or the OS kernel. For example, PU ( purge file ) maps
*-----< onto FmpPurge and BR maps directly onto the OS kernel command BR.
*-----< It is therfore possible to pass the CI command security check but
*-----< fail at a deeper level ( FMP or the kernel ). For a list of the
*-----< interdependencies see the RTE–A System Manager’s Manual.
*-----<
*-----< CI commands fall into one of the five following interdependencies.
*-----<
*-----< (1)   The command is really a program. There could be up to 3 levels
*-----<       of security that have to be passed in order for the command
*-----<       to be allowed. The 3 levels are,
*-----<
*-----<                – CI security check ( can the user issue the command
*-----<                  from CI ).
*-----<
*-----<                – Program security ( does the user have enough
*-----<                  capability to run the program that implements
*-----<                  the command ).
*-----<
*-----<                – FMP or OS kernel command ( does the user have
*-----<                  enough capability to call the FMP routine or issue
*-----<                  the kernel command that the CI command maps onto ).
*-----<
*-----< (2)   The CI command maps onto an FMP routine, therefore there are
```

```
*-----<       two levels of security involed.
*-----<
*-----<                – CI security check ( can the user issue the command
*-----<                  from CI ).
*-----<
*-----<                – FMP routines ( does the user have the capability
*-----<                  to call the FMP routine that CI is about to use ).
*-----<
*-----<
*-----< (3)   The CI command maps onto an OS kernel command, therefore there
*-----<       are two levels of security involed.
*-----<
*-----<                – CI security check ( can the user issue the command
*-----<                  from CI ).
*-----<
*-----<                – OS kernel command ( does the user have the capability
*-----<                  to use the OS kernel command that CI is about
*-----<                  to use ).
*-----<
*-----< (4)   The CI command is handled internally. In this case there is only
*-----<       one level of security involved, that of CI's.
*-----<
*-----< (5)   The CI command calls a documented utility routine ( as found
*-----<       in the Relocatable Library Reference Manual or the Programmer's
*-----<       Reference Manual. To find out which routines have security
*-----<       features in them see the $security: category above.
*-----<
*-----<

$category: ci

              alias  0   0   0   0   * alias commands            (4)
              as     0   0   0   0   * assign partition          (3)
              ask    0   0   0   0   * display promt/read response(1)
              at     10  0   0   0   * time schedule             (4)
              br     0   0   0   0   * Set break flag            (3)
              cd     0   0   0   0   * change directory          (2)
              cl     0   0   0   0   * list mounted disks        (4)
              cn     10  0   0   0   * device control            (4)
              co     0   0   0   0   * copy files                (2)
              cr     0   0   0   0   * create file               (2)
              crdir  0   0   0   0   * create directory          (2)
              cz     0   0   0   0   * code partition size       (3)
              dc     0   0   0   0   * dismount disk             (2)
              dl     0   0   0   0   * directory list            (1)
              dt     0   0   0   0   * data partition size       (3)
              echo   0   0   0   0   * echo parameters           (4)
              ex     0   0   0   0   * terminate CI              (4)
              fnctn  0   0   0   0   * create a function         (4)
              fnctns 0   0   0   0   * display functions         (4)
              go     0   0   0   0   * resume suspended program  (3)
              if     0   0   0   0   * control structure         (4)
              in     20  0   0   0   * initialize disk           (2)
              io     0   0   0   0   * display I/O config        (1)
              is     0   0   0   0   * control compare           (1)
              li     0   0   0   0   * list files                (1)
              mc     0   0   0   0   * mount disk                (2)
              mo     0   0   0   0   * move files                   (4)
              of     0   0   0   0   * stop prog                    (3)
              owner  0   0   0   0   * file ownership               (2)
              path   0   0   0   0   * UDSP control                 (1)
              poll   0   0   0   0   * async. CI cmd execution      (4)
              pr     0   0   0   0   * program priority             (3)
              prot   0   0   0   0   * file protection              (2)
              ps     0   0   0   0   * program status               (3)
              pu     0   0   0   0   * purge files                  (2)
              pwd    0   0   0   0   * display working directory    (4)
              return 0   0   0   0   * control return               (4)
              rn     0   0   0   0   * rename files                 (2)
              rp     10  0   0   0   * create id segment            (2)
              rs     0   0   0   0   * restart CI                   (1)
              ru     0   0   0   0   * run prog                     (2)
              set    8   0   0   0   * set user CI variable         (4)
              sp     0   0   0   0   * spool operation              (1)
              ss     0   0   0   0   * suspend program              (3)
              sz     0   0   0   0   * modify program size          (3)
              tm     0   0   0   0   * system time                  (3)
              to     12  0   0   0   * device time out              (4)
              tr     0   0   0   0   * transfere to cmd file        (4)
              ul     0   0   0   0   * unlock shareable EMA part.   (3)
              unalia 0   0   0   0   * clear an alias               (4)
              unpu   0   0   0   0   * unpurge file                 (2)
              unset  8   0   0   0   * clear user CI variable       (4)
              up     0   0   0   0   * up a device                  (3)
              vs     0   0   0   0   * ema/vma size                 (3)
              wd     0   0   0   0   * working directory            (2)
              wh     0   0   0   0   * system status                (1)
              while  0   0   0   0   * control structure            (4)
              whosd  0   0   0   0   * volume users                 (1)
              ws     0   0   0   0   * vma working set              (3)
              xq     0   0   0   0   * nowait prog run              (2)
              ?      0   0   0   0   * help                         (4)
*
*-----< The security category for GRUMP.
*
*
*-----<   Meanings of the base and subfunctions fall into one of these
*-----<   two categories.
*-----<
*-----<   (1)   Base    - Capability needed to perform the function on
*-----<                   yourself.
*-----<         Sub01   - Capability needed to perform the function on
*-----<                   someone else.
*-----<         Sub02   - Not defined.
*-----<         Sub03   - Not defined.
*-----<
*-----<   (2)   Base    - Capability needed to perform the function on
*-----<                   the group of which you are a member.
*-----<         Sub01   - Capability needed to perform the function on
*-----<                   the group of which you are not a member.
*-----<         Sub02   - Not defined.
*-----<         Sub03   - Not defined.
*
```

```
$category: grump

              algrln  31  31  0   0   *alter group logon name          (2)
              algrcp  25  31  0   0   *alter group cpu limit           (2)
              algrco  25  31  0   0   *alter group connect time limit  (2)
              algrbm  25  31  0   0   *alter group bit map             (2)
              alusln  31  31  0   0   *alter logon user name           (1)
              alusrn  10  31  0   0   *alter logon user real name      (1)
              aluspw  10  31  0   0   *alter user password             (1)
              alusud  25  31  0   0   *alter user udsp depth and levels(1)
              aluscl  31  31  0   0   *alter user cplv                 (1)
              alusbm  25  31  0   0   *alter user bit map              (1)
              aluscp  31  31  0   0   *alter user cpu limit            (1)
              alusco  31  31  0   0   *alter user connect limit        (1)
              alusgr  25  31  0   0   *alter user group list (add groups) (1)
              aludlg  15  31  0   0   *alter user default logon group  (1)
              alussu  15  31  0   0   *alter user startup program      (1)
              aluslf  15  31  0   0   *alter user logoff program       (1)
              aluswd  31  31  0   0   *alter user logon working dir.   (1)
              ligrp   15  31  0   0   *list group definitions          (2)
              liusr   15  31  0   0   *list user definitions           (1)
              passwd  0   25  0   0   *change user password            (1)
              alusen  25  31  0   0   *alter user EVB size             (1)
*
*-----<  The meanings of the base and subfunctions are now;
*-----<
*-----<   Base    – Capability needed to issue the command.
*-----<   Sub01   – Not defined.
*-----<   Sub02   – Not defined.
*-----<   Sub03   – Not defined.
*

              negrp   31  0   0   0   *create new group
              neusr   31  0   0   0   *create new user
              pugrp   31  0   0   0   *purge a group
              puusr   31  0   0   0   *purge a user
              regrp   31  0   0   0   *reset a groups accounting info
              reusr   31  0   0   0   *reset a users accounting info
              kilses  31  0   0   0   *kill a session

*
*-----< The utils category.
*

$category: utils

      kilses  31  0   0   0   *Kill a session, yes or on

*
*-----< Meanings of functions for seslu.
*-----<
*-----<   Base    – List your sessions’s bit map
*-----<   Sub01   – List another session’s bit map
*-----<   Sub02   – Modify your session’s bit map
*-----<   Sub03   – Modify another session’s bit map
*
```

```
      seslu   5   10  15  31  *SESLU utility


*
*-----< The following two categories (HP000 and HP001) are reserved
*-----< for use by HP.
*

$category: hp000
      res00  0  0  0  0
      res01  0  0  0  0
      res02  0  0  0  0
      res03  0  0  0  0
      res04  0  0  0  0
      res05  0  0  0  0
      res06  0  0  0  0
      res07  0  0  0  0
      res08  0  0  0  0
      res09  0  0  0  0
      res10  0  0  0  0
      res11  0  0  0  0
      res12  0  0  0  0
      res13  0  0  0  0
      res14  0  0  0  0
      res15  0  0  0  0
      res16  0  0  0  0
      res17  0  0  0  0
      res18  0  0  0  0
      res19  0  0  0  0

$category: hp001

      res00  0  0  0  0
      res01  0  0  0  0
      res02  0  0  0  0
      res03  0  0  0  0
      res04  0  0  0  0
      res05  0  0  0  0
      res06  0  0  0  0
      res07  0  0  0  0
      res08  0  0  0  0
      res09  0  0  0  0
      res10  0  0  0  0
      res11  0  0  0  0
      res12  0  0  0  0
      res13  0  0  0  0
      res14  0  0  0  0
      res15  0  0  0  0
      res16  0  0  0  0
      res17  0  0  0  0
      res18  0  0  0  0
      res19  0  0  0  0
```

E

Security Table Worksheet

This appendix contains tables that show the relationship of the $system and $fmp security functions to the CI commands, and the relationship of the $fmp security functions to FMP routines. There is also a worksheet (Table E-3) designed to aid in the creation and modification of your security table. This information can be used to track down the source of a security violation error for CI commands, by determining at which level (CI, OS, or FMP) the error occurred. The table also indicates if the command is subject to program security and file system checking.

The worksheet contains blanks where you can record the capability levels for each command and routine. For example, the AT command in CI has one level, the base (B). If you are using the security table supplied by Hewlett-Packard (SECURITY.TBL), the default capability level is already determined, and you would put a 10 there. The system DT command has four levels, the base and three subfunctions (B, 1, 2, 3). The default capability levels are 10 10 20 25. The definition of the levels of the commands and routines is contained in SECURITY.TBL (see Appendix D).

The worksheet is completed in the following manner. The CI AS command is used as an example. When assigning capability levels, you must be aware that the AS command is also checked at the OS command level. If your capability level is 11, the required capability for the CI AS command is set to 10, and the OS AS command level definition is 12 15 20 30, you will not be able to execute any of the functions of the AS command because you will fail the security check in the OS on all function levels. Thus, even though you passed the check at the CI level, you could not execute the command because you failed at the system level. If your cabability level is 16, you will be able to perform the base and first subfunction of the AS command, but not the second or third subfunction.

A different example is the CI IN command. The IN command calls the three FMP commands that correspond to the $fmp security functions listed in the worksheet. If the values are defined below, and your capability level is 20, you will get a security violation detected when the IN command attempts to do the FmpMount(init) because your level is not high enough. The following is an example of this condition:

CI Command	**$system Security Function**	**$fmp Security Function**
IN		init
B 20		B 25
		mount
		B 20
		dismnt
		B 20

The following table shows the relationship of the OS commands, and the FMP routines, to the CI commands, as well as indicating if the command is subject to program security and file system checking. The checks for the listed FMP routines are only performed if the CI command is issued in a manner that exercises the routine. For example, the LI command only uses FmpOpen(create) if you say to divert the listing to a file that does not exist.

Table E-1. Security Table Structure for CI Commands

CI Command	$system Security Function	$fmp Security Function	Required Program	File System
ALIAS	None	None	None	No
AS	as	None	None	No
ASK	None	None	ASK	Yes
AT	None	None	CIX, D.RTR Specified Program	Yes
BR	br	None	None	No
CD	None	wd setwd	None	Yes
CL	None	None	None	No
CN	None	None	None	No
CO	None	crdir open create info setdir purge	CIX, D.RTR	Yes
CR	None	create	CIX, D.RTR	Yes
CRDIR	None	crdir	CIX, D.RTR	Yes
CZ	cd	None	None	No
DC	None	dismnt	CIX, D.RTR	Yes
DL	None	open create wd opfile	DL	Yes
DT	dt	None	None	No

CI Command	$system Security Function	$fmp Security Function	Required Program	File System
ECHO	None	None	None	No
EX	of	open	CIALOGOF Logof Prog. LI	Yes
FUNCTION (fnctn)	None	None	None	No
FUNCTIONS (fnctns)	None	None	None	No
GO	go	None	None	No
IF–THEN–ELSE–FI (if)	None	None	None	No
IN	None	init mount dismnt	CIX, D.RTR	Yes
IO	None	create	IO	Yes
IS	None	None	IS	No
LI	None	open create reccnt purge	LI	Yes
MC	None	init mount	CIX, D.RTR	Yes
MO	None	rename	CIX, D.RTR	Yes
OF	of	None	None	No
OWNER	None	setown	CIX, D.RTR	Yes
PATH	None	open dirnam	PATH, D.RTR	No
POLL	None	None	None	No
PR	pr	None	None	No
PROT	None	prot	DL, CIX	Yes

CI Command	$system Security Function	$fmp Security Function	Required Program	File System
PS	ps	None	None	No
PU	None	purge	CIX, D.RTR	Yes
PWD	None	wd	None	Yes
RETURN	None	None	None	No
RN	None	rename	CIX, D.RTR	Yes
RP	None	None	CIX, D.RTR	Yes
RS	of	None	RS	Yes
RU	None	None	Specific Program	Yes
SET	None	None	None	No
SP	None	open create purge setprt crdir	SP	Yes
SS	ss	None	None	No
SZ	sz	None	None	No
TM	tm	None	CIX	No
TO	None	None	None	No
TOUCH	None	open create opfiles acctim cretim updtim setdir wd	TOUCH	Yes
TR	None	open	None	Yes
UL	ul	None	None	No
UNALIAS (unalia)	None	None	None	No

CI Command	$system Security Function	$fmp Security Function	Required Program	File System
UNPU	None	unprg	CIX, D.RTR	Yes
UNSET	None	None	None	No
UP	up	None	None	No
VS	vs	None	None	No
WC	None	info open create opfile wd	WC	Yes
WD	None	wd setwd open	None	Yes
WH	None	None	WH	No
WHILE–DO–DONE (while)	None	None	None	No
WHOSD	None	dirnam info open create opfile wd	WHOSD	Yes
WS	ws	None	None	No
XQ	None	None	Specific Program	Yes
?	None	None	LI	Yes

Table E-2. Security Table Structure for FMP Routines

FMP Routine	$fmp Security Routine
Calc_Dest_Name	None
DcbOpen	None
FattenMask	None
FmpAccessTime	acctim
FmpAppend	info
FmpBitBucket	None
FmpBuildHierarch	None
FmpBuildName	None
FmpBuildPath	None
FmpCloneName	None
FmpClose	None
FmpControl	None
FmpCopy	open create purge info setdir filenm rename trunct
FmpCreateDir	crdir
FmpCreateTime	cretim
FmpDcbPurge	None
FmpDevice	None
FmpDismount	dismnt
FmpEndMask	None
FmpEof	eof
FmpError	None
FmpExpandSize	None
FmpFileName	filenm
FmpFPos	None
FmpHierarchName	None

FMP Routine	$fmp Security Routine
FmpInfo	info
FmpInitMask	open wd
FmpInteractive	None
FmpIoOptions	None
FmpIoStatus	None
FmpLastFileName	None
FmpList	open create
FmpListX	open create
FmpLu	None
FmpMakeSLink	open create
FmpMaskName	None
FmpMount	mount
FmpNextMask	open opfile
FmpOpen	open create
FmpOpenFiles	opfile
FmpOpenScratch	open create filenm
FmpOpenTemp	None
FmpOwner	open
FmpPackSize	None
FmpPagedDevWrite	None
FmpPagedWrite	None
FmpPaginator	None
FmpParseName	None
FmpParsePath	None
FmpPosition	None
FmpPost	None

FMP Routine	$fmp Security Routine
FmpPostEof	None
FmpProtection	prot
FmpPurge	purge
FmpRawMove	None
FmpRead	None
FmpReadLink	open
FmpReadString	None
FmpRecordCount	reccnt
FmpRecordLen	reclen
FmpRename	rename
FmpReportError	None
FmpRewind	None
FmpRpProgram	open wd
FmpRunProgram	open wd
FmpRwBits	None
FmpSetDcbInfo	None
FmpSetDirInfo	setdir
FmpSetEof	None
FmpSetFPos	None
FmpSetIoOptions	None
FmpSetOwner	open setown
FmpSetPosition	None
FmpSetProtection	setprt
FmpSetWord	None
FmpSetWorkingDir	setwd
FmpShortName	filenm
FmpSize	size
FmpStandardName	None
FmpTruncate	trunct

FMP Routine	$fmp Security Routine
FmpUdspEntry	dirnam
FmpUdspInfo	None
FmpUniqueName	None
FmpUnPurge	unprg
FmpUpdateTime	updtim
FmpWorkingDir	wd
FmpWrite	None
FmpWriteString	None
MaskDiscLu	None
MaskIsDS	None
MaskMatchLevel	None
MaskOldFile	None
MaskOpenId	opfile
MaskOwnerIds	None
MaskSecurity	None
WildCardMask	None
DsCloseCon	None
DsDcbWord	None
DsDiscInfo	None
DsDiscRead	None
DsFstat	None
DsNodeNumber	None
DsOpenCon	None
DsSetDcbWord	None

The following worksheet can aid you in customizing your security table. Write the desired capability level in the blanks under each command or routine. A capability of 0 can be specified. For example, subfunctions 1, 2, 3 of FmpMount and FmpDismount are 0 in the Hewlett-Packard supplied SECURITY.TBL.

Table E-3. Security Table Worksheet

CI Command	$system Security Function	$fmp Security Function
ALIAS B ___		
AS B ___	as B ___ 1 ___ 2 ___ 3 ___	
ASK B ___		
AT B ___		
BR B ___	br B ___ 1 ___ 2 ___ 3 ___	
CD B ___		wd B ___ setwd B ___ 1 ___ 2 ___ 3 ___
CL B ___		
CN B ___		

CI **Command**	**$system** **Security Function**	**$fmp** **Security Function**
CO B ___		crdir B ___ 1 ___ 2 ___ 3 ___ open B ___ 1 ___ 2 ___ 3 ___ create B ___ 1 ___ 2 ___ 3 ___ info B ___ setdir B ___ purge B ___ 1 ___ 2 ___ 3 ___
CR B ___		create B ___ 1 ___ 2 ___ 3 ___
CRDIR B ___		crdir B ___ 1 ___ 2 ___ 3 ___
CZ B ___	cd B ___ 1 ___ 2 ___ 3 ___	
DC B ___		dismnt B ___ 1 ___ 2 ___ 3 ___
DL B ___		open B ___ 1 ___ 2 ___ 3 ___ create B ___ 1 ___ 2 ___ 3 ___ wd B ___ opfile B ___

CI Command	$system Security Function	$fmp Security Function
DT B ___	dt B ___ 1 ___ 2 ___ 3 ___	
ECHO B ___		
EX B ___	of B ___ 1 ___ 2 ___ 3 ___	open B ___ 1 ___ 2 ___ 3 ___
FUNCTION B ___		
FUNCTIONS B ___		
GO B ___	go B ___ 1 ___ 2 ___ 3 ___	
IF–THEN–ELSE–FI B ___		
IN B ___		init B ___ 1 ___ 2 ___ 3 ___ mount B ___ 1 ___ 2 ___ 3 ___ dismnt B ___ 1 ___ 2 ___ 3 ___
IO B ___		create B ___ 1 ___ 2 ___ 3 ___

CI Command	$system Security Function	$fmp Security Function
IS B ___		
LI B ___		open B ___ 1 ___ 2 ___ 3 ___ create B ___ 1 ___ 2 ___ 3 ___ reccnt B ___ purge B ___ 1 ___ 2 ___ 3 ___
MC B ___		init B ___ 1 ___ 2 ___ 3 ___ mount B ___ 1 ___ 2 ___ 3 ___
MO B ___		rename B ___ 1 ___ 2 ___ 3 ___
OF B ___	of B ___ 1 ___ 2 ___ 3 ___	
OWNER B ___		setown B ___ 1 ___ 2 ___ 3 ___
PATH B ___		open B ___ 1 ___ 2 ___ 3 ___ dirnam B ___

CI Command	$system Security Function	$fmp Security Function
POLL B ___		
PR B ___	pr B ___ 1 ___ 2 ___ 3 ___	
PROT B ___		prot B ___ 1 ___ 2 ___ 3 ___
PS B ___	ps B ___ 1 ___ 2 ___ 3 ___	
PU B ___		purge B ___ 1 ___ 2 ___ 3 ___
PWD B ___		wd B ___
RETURN B ___		
RN B ___		rename B ___ 1 ___ 2 ___ 3 ___
RP B ___		
RS B ___	of B ___ 1 ___ 2 ___ 3 ___	

CI Command	$system Security Function	$fmp Security Function
RU B ___		
SET B ___		
SP B ___		open B ___ 1 ___ 2 ___ 3 ___ create B ___ 1 ___ 2 ___ 3 ___ purge B ___ 1 ___ 2 ___ 3 ___ setprt B ___ crdir B ___ 1 ___ 2 ___ 3 ___
SS B ___	ss B ___ 1 ___ 2 ___ 3 ___	
SZ B ___	sz B ___ 1 ___ 2 ___ 3 ___	
TM B ___	tm B ___ 1 ___	
TO B ___		

CI Command	$system Security Function	$fmp Security Function
TOUCH B ___		open B ___ 1 ___ 2 ___ 3 ___ create B ___ 1 ___ 2 ___ 3 ___ opfiles B ___ acctim B ___ cretim B ___ updtim B ___ setdir B ___ wd B ___
TR B ___		open B ___ 1 ___ 2 ___ 3 ___
UL B ___	ul B ___	
UNALIAS B ___		
UNPU B ___		unprg B ___ 1 ___ 2 ___ 3 ___
UNSET B ___		
UP B ___	up B ___	

CI Command	$system Security Function	$fmp Security Function
VS B ___	vs B ___ 1 ___ 2 ___ 3 ___	
WC B ___		info B ___ open B ___ 1 ___ 2 ___ 3 ___ create B ___ 1 ___ 2 ___ 3 ___ opfile B ___ wd B ___
WD B ___		wd B ___ setwd B ___ 1 ___ 2 ___ 3 ___ open B ___ 1 ___ 2 ___ 3 ___
WH B ___		
WHILE-DO-DONE B ___		

CI Command	$system Security Function	$fmp Security Function
WHOSD B ___		dirnam B ___ info B ___ open B ___ 1 ___ 2 ___ 3 ___ create B ___ 1 ___ 2 ___ 3 ___ opfile B ___ wd B ___
WS B ___	ws B ___ 1 ___ 2 ___ 3 ___	
XQ B ___		
? B ___		

F

Security/1000 Error Codes

−1700 Stgen’s symbol table overflowed.

This is an internal error; contact Hewlett-Packard.

−1701 Security/1000 not generated into your system.

The OS modules SECOS.REL and CHECK.REL must be generated into your system in order to use Security/1000.

−1702 No such category or function.

The category or function specified cannot be found in the Security/1000 tables. Check your listing of the installed tables. (See the LT command in the SECTL utility.)

−1703 Security/1000 NOT turned on.

Certain Security/1000 functions can only be performed if Security/1000 is turned on. (See the IN and ON commands in the SECTL utility.)

−1704 Security/1000 has not been initialized.

Security/1000 must be initialized in order to turn on or use Security/1000. (See the IN command in the SECTL utility.)

−1705 Illegal value for CPLV, must be in the range 0-31.

If you are using the Security/1000 tables to store your application’s configuration information requiring values outside the 0-31 range, use the routines SecPutCitNam, SecPutCftNam, SecPutCitNum, and SecPutCftNum found in the SEC1000.LIB library.

−1706 **Duplicate category name.**

The category name already exists in the Security/1000 tables.

−1707 **Duplicate function name.**

The function name already exists within the specified category.

−1708 **Specified program NOT found.**

An ID segment belonging to the specified program cannot be found.

−1709 **Specified file not a SNAP file or corrupt SNAP file.**

The supplied file name was not a snap file. If it was a snap file, it was corrupt.

−1710 **STGEN found errors.**

The table generator, STGEN, found errors. See the list file produced by STGEN for the details.

−1711 **Renaming a category or function to <blank> not allowed.**

The category and function names cannot be blank in the Security/1000 tables.

−1712 **Reserved.**

−1713 **Security violation.**

The requested action cannot be allowed because you do not have the required capability level.

G

Security/1000 Library Routines

The Security/1000 Library Routines (SEC1000.LIB) are the programmatic interface of the Security/1000 subsystem. The user level subroutines are listed in Table G-1, and are described in this appendix in alphabetical order. Category and function name parameters are not modified by the programmatic security routines. Thus, you can define category and function names to be any 3-word value. However, any category or function name that is not upshifted, blank-filled, and left-justified cannot be accessed with SECTL. Thus, category and function names that are not upshifted, blank-filled, and left-justified must be defined, accessed, and altered with these routines.

Note that when a parameter is an entry number, it is always zero relative.

Table G-1. SEC1000.LIB Commands

Command	Purpose	Page #
SecChangeCplv	Changes PROGCPLV	G-3
SecChkCplvNam	Checks CPLV by name	G-3
SecChkCplvNum	Checks CPLV by number	G-4
SecEditFunction	Changes function's CPLV	G-4
SecGenTables	Generates security tables	G-5
SecGetCitNam	Gets CIT entry by name	G-5
SecGetCftNam	Gets CFT entry by name	G-6
SecGetCitNum	Gets CIT entry by number	G-6
SecGetCftNum	Gets CFT entry by number	G-7
SecGetCplvs	Gets CPLVs related to calling program	G-7
SecGetMyCplv	Gets calling program's CPLV word	G-7
SecGetRqusCplv	Gets RQUSCPLV of a program	G-8
SecGetProgCplv	Gets PROGCPLV of a program	G-8
SecInitialize	Initializes Security/1000	G-9
SecListTables	Lists security tables	G-9
SecOnOff	Determines if security is on/off	G-9
SecProgCplv	Gets program's CPLVs	G-10
SecPutCitNam	Updates CIT entry by name	G-10
SecPutCftNam	Updates CFT entry by name	G-11
SecPutCitNum	Updates CIT entry by number	G-11
SecPutCftNum	Updates CFT entry by number	G-12
SecPutRqusCplv	Sets RQUSCPLV of a program	G-12
SecPutProgCplv	Sets PROGCPLV of a program	G-13
SecRenameCat	Renames a category	G-13
SecRenameFnc	Renames a function	G-14
SecSwitch	Turns security on/off	G-14
SecUserCplv	Gets USERCPLV	G-15

SecChangeCplv

SecChangeCplv allows the calling program to change its PROGCPLV during execution.

```
call SecChangeCplv(newcplv,error)
error=SecChangeCplv( newcplv,error)
integer newcplv,error
```

newcplv An integer specifying the new PROGCPLV. It ranges from zero to the maximum of USERCPLV and ORGCPLV. If *newcplv* is negative, PROGCPLV is set to ORGCPLV.

error An integer that returns a negative value if there is an error.

SecChkCplvNam

SecChkCplvNam checks a given CPLV against the four capability levels of a function to determine whether the CPLV equals or exceeds each level of the function (via function and category names).

```
call SecChkCplvNam(fncnam,catnam,chkcplv,flagarray,error)
error=SecChkCplvNam( fncnam,catnam,chkcplv,flagarray,error)
integer fncnam(3),catnam(3),flagarray(4)
integer chkcplv,error
```

fncnam A 3-word integer array containing the function name whose CPLVs are checked against the supplied CPLV. The function name must be left-justified, blank-filled, and upshifted as needed.

catnam A 3-word integer array containing the name of the category owning the function in the *fncnam* parameter. The category name must be left-justified, blank-filled, and upshifted as needed.

chkcplv An integer checked against the function's capability levels.

flagarray A 4-word integer array containing the CPLV flags. Each flag indicates whether the *chkcplv* was less than the CPLV in the security tables. If a flag is <0, *chkcplv* was less than the corresponding CPLV in the security table; if the flag is >0, *chkcplv* was equal to or greater than the corresponding security table CPLV. Word 1 = base CPLV, Word 2 = sub1 CPLV, Word 3 = sub2 CPLV, Word 4 = sub3 CPLV.

If the function cannot be found, *flagarray* is set to positive values. ERROR is then set to positive 1702. This indicates that the test passed because the function could not be found. This is not considered an error condition, but enables a programmer to detect the situation.

error An integer that returns a negative value if there is an error. Note that if a positive 1702 is returned, then the function could not be found. This is not considered an error condition.

If the function has less than 3 subfunctions, the flags of the missing subfunctions are set to a positive value.

SecChkCplvNam differs from SecChkCplvNum in that SecChkCplvNam searches the tables for a particular name while SecChkCplvNum searches for a particular number.

SecChkCplvNum

SecChkCplvNum checks a given CPLV against the four capability levels of a given function to determine whether it equals or exceeds each level of that function (via function and category numbers).

```
call SecChkCplvNum(fncnum,catnum,chkcplv,flagarray,error)
error=SecChkCplvNum(fncnum,catnum,chkcplv,flagarray,error)
integer fncnum,catnum,flagarray(4)
integer chkcplv,error
```

fncnum An integer that specifies the number of the CFT entry to be used in the CPLV check.

catnum An integer that specifies the numbner of the CIT entry that points to the CFT to be used with the *fncnum* parameter.

chkcplv An integer that specifies the CPLV to be checked against the function's capability levels.

flagarray A 4-word integer array that returns the CPLV flags. Each flag indicates whether *chkcplv* was less than the CPLV in the security tables. If the flag is <0, *chkcplv* is less than corresponding security table CPLV; if the flag is >0, *chkcplv* is equal to or greater than security table CPLV. Word 1 = base CPLV, Word 2 = sub1 CPLV, Word 3 = sub2 CPLV, Word 4 = sub3 CPLV.

If the function cannot be found, *flagarray* is set to positive values. ERROR is then set to positive 1702. This indicates that the test passed because the function could not be found. This is not considered an error condition, but enables a programmer to detect the situation.

error An integer that returns a negative value if there is an error. Note that if a positive 1702 is returned, then the function could not be found. This is not considered an error condition.

If the function has less than 3 subfunctions, the flags of the undefined subfunctions are set to a positive value.

SecEditFunction

SecEditFunction changes the CPLVs for a specified function.

```
call SecEditFunction(fncnam,catnam,cplvarray,error)
integer catnam(3),fncnam(3),cplvarray(4)
integer error
```

catnam A 3-word integer array containing the category name of the specified function. The contents of the array must be left-justified, blank-filled, and upshifted as needed.

fncnam A 3-word integer array containing the function name in the specified category whose CPLV will be changed. The array must be left-justified, blank-filled, and upshifted as needed.

cplvarray	A 4-word integer array containing the new CPLV values. Word 1 = new base CPLV, Word 2 = new sub1 CPLV, Word 3 = new sub2 CPLV, Word 4 = new sub3 CPLV. All CPLVs are in the 0-31 range. If a CPLV is −1, the corresponding CPLV in the CFT entry will not be updated.
error	An integer that returns a negative value if there is an error.

SecGenTables

SecGenTables generates a set of security tables.

call SecGenTables(*inputfile*,*listfile*,*outputfile*,*keepmac*,*error*)
character*64 *inputfile*,*listfile*,*outputfile*,*keepmac*
integer *error*

inputfile	A character string that specifies the name of the file containing the source of the security tables.
listfile	A character string that specifies the file to which the listing will be written. The file will be created if it does not exist, overwritten if it does. If it is blank or ASCII 0, the listing is not saved.
outputfile	A character string that specifies the file to which the generated security tables will be returned. The file will be created if it does not exist, overwritten if it does.
keepmac	A character string that specifies the name of the file to which the generated MACRO/1000 code will be written. The file will be created if it does not exist, overwritten if it does. If it is a blank or ASCII 0, the generated Macro/1000 code is not saved.
error	An integer that returns a negative if there is an error. All FMP error codes have been biased so you can determine the file to which the error code relates. The bias factors are: SOURCE −2000, LIST −3000, OUTPUT −4000, KEEPMAC −5000. To get original error code: FmpError = error + abs(bias).

SecGetCitNam

SecGetCitNam retrieves an entry from the CIT via a category name.

call SecGetCitNam(*catnam*,*entry*,*number*,*address*,*error*)
error=SecGetCitNam(*catnam*,*entry*,*number*,*address*,*error*)
integer *catnam*(3),*entry*(4)
integer *number*,*address*,*error*

catnam	A 3-word integer array containing the name of the CIT entry to be retrieved. The category name must be left-justified, blank-filled, and upshifted as needed.
entry	A 4-word integer array that returns the CIT entry.
number	An integer that returns the number of the entry within the CIT (zero relative).
address	An integer that returns the the address of the CIT entry.
error	An integer that returns a negative value if there is an error.

SecGetCftNam

SecGetCftNam retrieves an entry from a CFT via a function and category name.

```
call SecGetCftNam(fncnam,catnam,entry,number,address,error)
error=SecGetCftNam( fncnam,catnam,entry,number,address,error)
integer fncnam(3),catnam(3),entry(7)
integer number,address,error
```

fncnam	A 3-word integer array containing the function name of the CFT entry to be returned. The function name must be left-justified, blank-filled, and upshifted as needed.
catnam	A 3-word integer array containing the name of the category owning the CFT from which the entry will be returned. The category name must be left-justified, blank-filled, and upshifted as needed.
entry	A 7-word integer array that returns the CFT entry.
number	An integer that returns the number of the entry within the CFT (zero relative).
address	An integer that returns the address of the CFT entry.
error	An integer that returns a negative value if there is an error.

SecGetCitNum

SecGetCitNum retrieves an entry from the CIT via an entry number.

```
call SecGetCitNum(catnum,entry,addr,error)
error=SecGetCitNum( catnum,entry,addr,error)
integer catnum,address,error,entry(4)
```

catnum	An integer that specifies the number of the CIT entry to be retrieved. Entries start numbering at zero.
entry	A 4-word integer array that returns the CIT entry.
address	An integer that returns the address of the CIT entry.
error	An integer that returns a negative value if there is an error.

SecGetCftNum

SecGetCftNum retrieves an entry from a CFT via CIT and CFT entry numbers.

`call SecGetCftNum(`*fncnum*`,`*catnum*`,`*entry*`,`*address*`,`*error*`)`
error`=SecGetCftNum(`*fncnum*`,`*catnum*`,`*entry*`,`*address*`,`*error*`)`
`integer` *fncnum*`,`*catnum*`,`*address*`,`*error*`,`*entry*`(7)`

fncnum	An integer that specifies the number of the CFT entry to be retrieved. CFT entries number starting from zero.
catnum	An integer that specifies the number of the CIT entry that points to the CFT to be used with the *fncnum* parameter.
entry	A 7-word integer array that returns the CFT entry.
address	An integer that returns the address of the CFT entry.
error	An integer that returns a negative value if there is an error.

SecGetCplvs

SecGetCplvs retrieves the CPLVs of the calling program.

`call SecGetCplvs(`*usercplv*`,`*progcplv*`,`*rquscplv*`,`*orgcplv*`)`
`integer` *usercplv*`,`*progcplv*`,`*rquscplv*`,`*orgcplv*

usercplv	An integer that returns the USERCPLV of the calling program.
progcplv	An integer that returns the PROGCPLV of the calling program.
rquscplv	An integer that returns the RQUSCPLV of the calling program.
orgcplv	An integer that returns the ORGCPLV of the calling program.

SecGetMyCplv

SecGetMyCplv retrieves the CPLV word from the caller's ID segment extension.

cplv`=SecGetMyCplv()`
`integer` *cplv*

cplv	An integer that returns the CPLV word from the caller's ID segment extension. Note that this is the CPLV word and *not* the cplv field within the CPLV word.

SecGetRqusCplv

SecGetRqusCplv retrieves the RQUSCPLV of the specified program.

```
call SecGetRqusCplv(progname,session,rquscplv)
error=SecGetRqUsCplv(progname,session,rquscplv)
integer progname(3)
integer session,rquscplv
```

progname	A 3-word integer array containing the program name, left-justified, blank-filled, and upshifted, of the ID segment whose RQUSCPLV field is to be retrieved. If the first word of *progname* is 0, the calling program is assumed to be the program specified.
session	An integer that specifies the session number in which the program resides. There are 2 special values for session: −1 = calling program's session, 0 = system session.
rquscplv	An integer that returns the RQUSCPLV field from the ID segment extension. It is a negative value if there is an error.
error	An integer that returns a negative value if there is an error.

SecGetProgCplv

SecGetProgCplv retrieves the PROGCPLV of the specified program.

```
call SecGetProgCplv(progname,session,rquscplv)
error= SecGetProgCplv(progname,session,progcplv)
integer progname(3)
integer session,progcplv,error
```

progname	A 3-word integer array containing the program name whose RQUSCPLV field is to be retrieved. The program name must be left-justified, blank-filled, and upshifted as needed. If the first word of *progname* is 0, the calling program is assumed to be the program specified.
session	An integer that specifies the session number in which the program resides. There are 2 special values for session: −1 = calling program's session, 0 = system session.
progcplv	An integer that returns the PROGCPLV field from the ID segment extension. It is a negative value if there is an error.

If there was no error, a non-negative value was returned, and the B-Register contains the address of the CPLV in the ID segment.

SecInitialize

SecInitialize initializes Security/1000, and turns it on as an option.

```
call SecInitialize(snapfile,onoff,error)
character*(*) snapfile
integer onoff,error
```

snapfile	A character string that specifies the name of the snapfile created at generation time and used by the current system.
onoff	An integer flag that specifies ON (non-zero) or OFF (zero).
error	An integer that returns a negative value if there is an error.

Security/1000 should be initialized and turned on only once. This should be done at system boot time before any users are allowed on the system.

Caution If a snapfile different from the current system snapfile is specified, your system and security tables can be corrupted.

SecListTables

SecListTables produces a listing of current installed security tables.

```
call SecListTables(listfile,error)
character*(*) listfile
integer error
```

listfile	A character string that specifies the file to which the listing will be written. The file is created if it does not exist, overwritten if it does.
error	An integer that returns a negative value if there is an error.

SecOnOf

SecOnOf determines whether Security/1000 is on or off.

```
flag=SecOnOf(flag) or
flag=SecOnOff(flag) or
flag=SecOnOf() or
flag=SecOnOff()
```

flag	An integer that returns a value indicating whether Security/1000 is on or off. Flag = 0 (OFF) or flag = non-zero (ON).

SecProgCplv

SecProgCplv retrieves the 3 fields from the CPLV word in the calling program's ID segment extension. If *idsegadr* is supplied, the information is retrieved from the specified ID segment's extension.

```
call SecProgCplv(orgcplv,rquscplv,curcplv[,idsegadr])
integer orgcplv,rquscplv,curcplv,idsegadr
```

orgcplv	An integer that returns the original program CPLV.
rquscplv	An integer that returns the required user CPLV for the program.
curcplv	An integer that returns the program's current CPLV.
idsegadr	An integer that specifies the ID segment address of the program whose CPLV word values are to be retrived. If it is not specified, the values are retrieved from the calling program's ID segment extension.

No error checking is done on the ID segment address, which is assumed to be correct.

SecPutCitNam

SecPutCitName updates an entry in the CIT via a category name.

Caution This routine does not check whether or not the updated entry causes duplicate categories. Incorrect use of this routine corrupts the CIT.

```
call SecPutCitNam(catnam,entry,number,address,error)
error=SecPutCitNam( catnam,entry,number,address,error)
integer catnam(3),entry(4)
integer number,address,error
```

catnam	A 3-word integer array containing the category name, left-justified, blank-filled, and upshifted as needed, of the category whose entry is to be updated.
entry	A 4-word integer array containing the updated version of the entry.
number	An integer that returns the number of the entry within the CIT (zero relative).
address	An integer that returns the address of the entry within the CIT.
error	An integer that returns a negative value if there is an error.

SecPutCftNam

SecPutCftNam updates an entry in a CFT.

Caution This routine does not check whether or not the updated entry causes duplicate functions. Incorrect use of this routine corrupts the CFT.

```
call SecPutCftNam(fncnam,catnam,entry,number,address,error)
error=SecPutCftNam(fncnam,catnam,entry,number,address,error)
integer fncnam(3),catnam(3),entry(7)
integer number,address,error
```

fncnam	A 3-word integer array containing the function name, left-justified, blank-filled, and upshifted as needed, whose entry is to be updated.
catnam	A 3-word integer array containing the name of the category, left-justified, blank-filled, and upshifted as needed, of the category owning the CFT to which the entry will be written.
entry	A 7-word integer array containing the updated version of the entry.
number	An integer that returns the number of the entry within the CFT (zero relative).
address	An integer that returns the address of the entry within the CFT.
error	An integer that returns a negative value if there is an error.

Note that this routine does not check whether or not the updated entry causes duplicate categories or functions.

SecPutCitNum

SecPutCitNum puts an updated entry back in the CIT via an entry number.

Caution This routine does not check whether or not the updated entry causes duplicate categories. Incorrect use of this routine corrupts the CIT.

```
call SecPutCitNum(catnum,entry,address,error)
error=SecPutCitNum( catnum,entry,address,error)
integer catnum,address,error,entry(4)
```

catnum	An integer that specifies the number of the CIT entry to be updated. CIT entries number starting from zero.
entry	A 4-word integer array for the buffer containing the updated version of the entry.
address	An integer that returns the address of the entry within the CIT.
error	An integer that returns a negative value if there is an error.

SecPutCftNum

SecPutCftNum updates and returns the CFT entry to its relevant CFT via the CIT and CFT entry numbers.

Caution This routine does not check whether or not the updated entry causes duplicate functions. Incorrect use of this routine corrupts the CFT.

```
call SecPutCftNum(fncnum,catnum,entry,address,error)
error=SecPutCftNum(fncnum,catnum,entry,address,error)
integer fncnum,catnum,address,error,entry(7)
```

fncnum	An integer that specifies the number of the CFT entry to be updated. CFT entries number starting from zero.
catnum	An integer that specifies the number of the CIT entry that points to the CFT to be indexed by *fncnum*. CIT entries numbers start from zero.
entry	A 7-word integer array for the buffer containing the updated entry version.
address	An integer that returns the address of the entry within the CFT.
error	An integer that returns a negative value if there is an error.

SecPutRqusCplv

SecPutRqusCplv sets the RQUSCPLV of a program.

```
error=SecPutRqusCplv(progname,session,newrquscplv,error)
integer progname(3)
integer session,newrquscplv,error
```

progname	A 3-word integer array containing the program name, left-justified, blank-filled. The program must have an ID segment set up. If the first word of *progname* is zero, the calling program's name is used.
session	An integer that specifies the session number in which the program resides. If the number is negative, use the calling program's session; if 0, use the system session.
newrquscplv	An integer that specifies the new RQUSCPLV value. This value cannot exceed the PROGCPLV of the program from which SecRqusCplv was called. Note that only the 5 least significant bits are looked at.
error	An integer that returns a negative value if there is an error.

SecPutProgCplv

SecPutProgCplv sets the PROGCPLV of a program.

```
error=SecPutProgCplv(progname,session,newprogcplv,error)
integer progname(3)
integer session,newprogcplv,error
```

progname	A 3-word integer array containing the program name, left-justified, blank-filled. The program must have an ID segment set up. If the first word of *progname* is zero, the calling program's name is used.
session	An integer that specifies the session number in which the program resides. If the number is negative, use the calling program's session, if 0, the system session.
newprogcplv	An integer that specifies is the new PROGCPLV value. This value cannot exceed the PROGCPLV of the program from which SecProgCplv was called. Note that only the 5 least significant bits are looked at.
error	An integer that returns a negative value if there is an error.

SecRenameCat

SecRenameCat renames a category name in the CIT. The old category must exist and the new one must not. Note that this routine does not modify the case (upper or lower) of the parameters. It is the caller's responsibility to ensure that the data passed to this routine is in the correct case.

Caution This routine does not check whether or not the updated entry causes duplicate categories. Incorrect use of this routine corrupts the CIT.

```
call SecRenameCat(oldcat,newcat,error)
integer oldcat(3),newcat(3)
integer error
```

oldcat	A 3-word integer array containing the old category name, left-justified, blank-filled, and upshifted as needed, which MUST exist in the CIT.
newcat	A 3-word integer array containing the new category name, left-justified, blank-filled, and upshifted as needed which, MUST NOT exist in the CIT.
error	An integer that returns a negative value if there is an error.

SecRenameFnc

SecRenameFnc renames a specified function in a specified category. Note that this routine does not modify the case (upper or lower) of the parameter. It is the caller's responsibility to ensure that the data passed to this routine is in the correct case.

Caution This routine does not check whether or not the updated entry causes duplicate functions. Incorrect use of this routine corrupts the CFT.

```
call SecRenameFnc(oldfnc,newfnc,cat,error)
integer oldfnc(3),newfnc(3),catnam
integer error
```

oldfnc	A 3-word integer array containing the old function name, left-justified blank-filled, and upshifted as needed, which MUST exist in the CFT.
newfnc	An integer array containing the new function name, left-justified, blank-filled, and upshifted as needed, which MUST NOT exist in the CFT.
catnam	A 3-word integer array containing the category name owning the function about to be renamed. The *catnam* parameter must be defined in the CIT, left-justified, blank-filled, and upshifted as needed.
error	An integer that returns a negative value if there is an error.

SecSwitch

SecSwitch switches Security/1000 on or off.

```
call SecSwitch(switch,error)
integer switch,error
```

switch	An integer that specifies the switch directive: 0 = off, non−zero = on.
error	An integer that returns a negative value if there is an error.

This routine can only be used after Security/1000 has been initialized. Security/1000 should be initialized and turned on only once at system boot up before any users are allowed on the system. Use of this routine is governed by the required CPLV defined for the switch function in the $SECURITY category in the security table whether Security/1000 is on or off.

Caution This routine was designed for system engineer debugging and should not be used to turn security on and off after users have been allowed on the system. Otherwise, system security cannot be guaranteed.

SecUserCplv

SecUserCplv retrieves the USERCPLV of the session in which the calling program resides.

UserCplv`=SecUserCplv()`
`Integer` *UserCplv*

UserCplv	An integer that is the USERCPLV.

H

Setup and Directory Create Programs

Example of a Setup Program

```
program setup
implicit none

*
*-----< This program renames or edits entries in the security tables.
*-----< It receives command input from a file whose name is supplied
*-----< in the runstring.
*-----<
*-----< An example of the command input to this program is given below.
*-----<
*-----<    rn,c,hp000,crdgp
*-----<    rn,f,crdgp,res00,nogroup
*-----<    rn,f,crdgp,res01,system
*-----<    ec,crdgp,system,0,1,2,3
*-----<    ex
*-----<
*-----< The command input format is the same as that accepted by the
*-----< SECTL utility.
*-----<
*-----< This program will work only if it has a high enough capability
*-----< level. The SECURITY/1000 routines will check the capability
*-----< level of this program to determine whether the requested action
*-----< can be allowed.
*

      integer indcb(144),runarray(40),xlog
      integer FmpOpen,error,DecimalToInt,length
      integer inarray(40),j
      integer CatArray(3),OldCatArray(3),NewCatArray(3)
      integer OldFncArray(3),NewFncArray(3),FncArray(3)
      integer CplvArray(4),fmpread

      character runstring*80,instring*80,command*6,cplvvalue*6
      character CategoryName*6,OldCategoryName*6,NewCategoryName*6
      character OldFunctionName*6,NewFunctionName*6,FunctionName*6

      equivalence (runstring,runarray)
      equivalence (instring,inarray)

      equivalence (OldFunctionName,  OldFncArray),
     +            (NewFunctionName,  NewFncArray),
     +            (OldCategoryName,  OldCatArray),
     +            (NewCategoryName,  NewCatArray),
     +            (CategoryName,     CatArray),
     +            (FunctionName,     FncArray)


      instring=' '


*
*-----< Pick up the runstring to find the name of the input file.
*

      call getst(runarray,-80,xlog)
      if(xlog.lt.80)runstring(xlog+1:)=' '

*
*-----< Open the input file.
*

      error=fmpopen(indcb,error,runstring,'or',1)
      if(error.lt.0)go to 9000

*
*-----< Start processing the commands.
*

10    length=Fmpread(indcb,error,inarray,80)
      if(length.eq.-1)go to 10000          ! EOF condition
      if(error.lt.0)go to 9000

*
*-----< Ensure that everything is in uppercase.
*

      call casefold(instring)

*
*-----< Get the command from the input string.
*

      call SplitString(instring,command,instring)

*
*-----< Find out what command was issued.
*-----< Commands are: RN - rename a category or function
*-----<               EC - edit a function within a category
*-----<               EX - exit
*



*
*-----< RN command. It has two subcommands.
*-----<               F - Rename a function
*-----<               C - Rename a category
*

      if(command(1:2).eq.'RN')then
          call SplitString(instring,command,instring)
          if(command(1:1).eq.'C')then
```

```
*
*-----< We will rename a category. Get the current name and the
*-----< new name.
*
              call Splitstring(instring,OldCategoryName,instring)
              call Splitstring(instring,NewCategoryName,instring)

*
*-----< Use the SECURITY/1000 routine SecRenameCat to perform the
*-----< rename.
*

              call SecRenameCat(OldCatArray,NewCatArray,error)
              if(error.lt.0)go to 8000
              go to 10                !go get next command
          endif

*
*-----< Check whether or not we are to rename a function.  If not, give
*-----< up and get the next command from the input file.
*


          if(command(1:1).eq.'F')then

*
*-----< We are to rename a function. Get the name of the category
*-----< that contains the function to be renamed.
*

              call SplitString(instring,CategoryName,instring)

*
*------< Get the current function name and the new name for the
*------< function.
*

              call Splitstring(instring,OldFunctionName,instring)
              call Splitstring(instring,NewFunctionName,instring)

*
*-----< Use the  SECURITY/1000 routine SecRenameFnc to perform the
*-----< function rename.
*

              call SecRenameFnc(OldFncArray,NewFncArray,
     +                          CatArray,error)
              if(error.lt.0)go to 8000
          endif
          go to 10    !go get next command
      endif


      if(command(1:2).eq.'EC')then

*
*-----< We will edit a function. Get the category containing the
*-----< function and the function to be edited.
*
```

```
          call SplitString(instring,CategoryName,instring)
          call SplitString(instring,FunctionName,instring)

*
*-----< Get the 4 CPLVs and convert them to binary. If an error
*-----< results during the conversion, the corresponding CPLV will
*-----< default to –1. A cplv value of –1 tells SECURITY/1000 not to
*-----< update the corresponding CPLV already in the tables.
*

          do j=1,4
              call SplitString(instring,CplvValue,instring)
              cplvarray(j)=DecimalToInt(CplvValue,error)
              if(error.ne.0)CplvArray(j)=–1
          enddo

*
*-----< Use the SECURITY/1000 routine SecEditFunction to perform
*-----< the actual editing operation.
*

          call SecEditFunction(FncArray,CatArray,
     +                         cplvarray,error)
          if(error.lt.0)go to 8000
          go to 10
      endif

*
*-----< See if an Exit command was issued.
*

      if(command(1:2).eq.’EX’)go to 10000


*
*-----< We receive this message if an unrecognized command was seen.
*

      write(1,1000)command
1000  format(” Unrecognized command seen ”,a6)
      go to 10

*
*-----< We have a SECURITY/1000 error.   Report it and terminate.
*

8000  write(1,8001)error
8001  format(” Security/1000 error: ”,i6.6)
      go to 10000

*
*-----< We have an Fmp error.   Report it and terminate.
*

9000  call FmpreportError(error,runstring)

10000 call FmpClose(indcb,error)
      end
```

Example of a Directory Create Program

```
program mkdir
implicit none

*
*-----< This program creates FMP directories. The program subjects the
*-----< user to a series of security checks before it will create
*-----< directories.
*-----<
*-----< The first check. Can the group with which the user logged on
*-----< create directories? The user’s logon group is checked to see
*-----< whether it is in the category ’CRDGP’. If it is, this test
*-----< is passed and test two is applied.
*-----<
*-----< The second check. Does the user have enough capability to create
*-----< a directory? The base function of the function whose name
*-----< matches that of the logon group is checked.
*-----<
*-----< If both tests are passed, control is passed to FmpCreateDir to
*-----< perform the actual directory creation. Note that FmpCreateDir
*-----< will perform its own security check as defined in the security
*-----< tables. See the SECURITY.TBL file for the security definition
*-----< of FmpCreateDir.
*

      integer runarray(40),xlog,GroupArray(5)
      integer SecGetMyCplv,ProgCplv,FlagArray(4),error
      integer DecimalToInt,lu,FmpCreateDir

      character runstring*80,GroupName*10,DirectoryName*64
      character LuString*4

      equivalence (runstring,runarray)
      equivalence (GroupName,grouparray)

*
*-----< Collect the run string to find out what the user wants to do.
*

      call getst(runarray,–80,xlog)
      if(xlog.lt.80)runstring(xlog+1:)=’ ’

*
*-----< Get the directory name and LU, if suppiled.


      call SplitString(runstring,DirectoryName,runstring)

*
*-----< See if a directory was supplied. If not, issue a little help.
*

      if(DirectoryName(1:1).eq.’ ’)then
          write(1,100)
100       format(” MkDir Usage: MkDir directory[,lu]”)
          call exec(6,0)
      endif
```

```
*
*-----< Get the LU, if supplied.
*

      call SplitString(runstring,LuString,runstring)
      if(LuString(1:1).ne.' ')then
          lu=DecimalToInt(LuString,error)
          if(error.ne.0)then
              write(1,100)
              call exec(6,0)
          endif
          else
          lu=0
      endif

*
*-----< Get my capability level. It will be needed for the security
*-----< check.
*

      ProgCplv=SecGetMyCplv()

*
*-----< Get the name of the group with which the user logged on.
*-----< Use the group name as a function name within category "CRDGP" to
*-----< determine what (if anything) the user can do with this utility.
*

      call GpNam(GroupName)

*
*-----< Now find out what the user can do with us.
*-----< Note that all errors are treated as security violations.
*-----< Also, if SecChkCplvNam is supplied with a category or function
*-----< that is not defined, it will return with the FlagArray set to
*-----< indicate that the check passed all functions (base and the three
*-----< subfunctions). It will also set error to POSITIVE 1702. This
*-----< indicates that the category or function could not be found, so
*-----< a default result, successful, was returned.  Failure to find the
*-----< category or function is NOT considered an error condition by
*-----< SecChkCplvNam.
*

      call SecChkCplvNam(GroupArray,6hCRDGP ,ProgCplv,FlagArray,error)
      if(error.lt.0)then
          write(1,1)
1         format(" SECURITY VIOLATION detected.")
          call exec(6,0)
          else
          if(error.eq.1702)then
              write(1,2)
2             format(" MkDir: No access for your group")
              call exec(6,0)
          endif
      endif

*
*-----< We know now that the user's group can create directories. But
*-----< whether the user can create directories can be determined by
*-----< looking at the flag corresponding to the base function.
```

```
*

      if(FlagArray(1).lt.0)then
          write(1,1)
          call Exec(6,0)
      endif


*
*-----< The type of directory the user can create will be determined
*-----< by the CRDIR function of the $FMP category. The FMP subsystem
*-----< will do this checking for us when we call the  FmpCreateDir
*-----< routine.
*

      error=FmpCreateDir(DirectoryName,lu)
      if(error.lt.0)then
          call FmpreportError(error,DirectoryName)
      endif


      end
```

RINFO and SINFO Utilities

Reset Multiuser Accounting Information (RINFO)

RINFO resets the multiuser accounting information found in the user configuration file for the specified users. It resets the multiuser accounting information to zero for cumulative connect time (words 95-96 in record one and words 11-12 in the USER.NOGROUP record) and cumulative CPU usage (words 97-98 in record one and words 9-10 in the USER.NOGROUP record).

Note RINFO was replaced by the RE (Reset) command in GRUMP. For situations where groups are not used, RINFO was updated to use the new multiuser data structures and function as it did before.

Calling RINFO

To call RINFO, enter the following runstring:

```
    CI> [RU] RINFO [UserMask1 [UserMask2[...]]]
```

The *UserMask* parameter is the name or file mask for users whose accounting information you want to set to 0. The default is the current user.

Loading RINFO

To load RINFO, use the following LINK command:

```
    CI> link,#rinfo
```

RINFO Protection

If SECURITY/1000 is turned on, the system manager can assign a required capability level to run RINFO; otherwise, only superusers can run RINFO.

Returned Values

RINFO returns the following 5 values through a PRTN call:

Word 1	Status (0=successful, -1=unsuccessful).
Words 2,3	Connect time in seconds for last user. Word 2 is the more significant word of the double integer value.
Words 4,5	CPU usage in tens of milliseconds for the last user. Word 4 is the more significant word of the double integer value.

Show Multiuser Accounting Information (SINFO)

SINFO displays the multiuser accounting information found in the user configuration file for the specified user. The information includes the last logoff time, cumulative connect time, and cumulative CPU usage.

Note SINFO was replaced by the LI (List) command in GRUMP. In situations where groups are not being used, or if you want the unique user information, SINFO retrieves information from record one of the user configuration file.

Calling SINFO

To call SINFO, enter the following runstring:

```
CI> [RU] SINFO [UserMask1 [UserMask2 [...]]]
```

The *UserMask* parameter is the name or file mask for the users whose accounting information you want displayed. The default is the current user.

Loading SINFO

To load SINFO, use the following LINK command:

```
CI> link,#sinfo
```

Returned Values

SINFO returns the following 5 values through a PRTN call:

Word 1	Status (0 = successful, -1 = unsuccessful).
Words 2,3	Connect time in seconds for last user. Word 2 is the more significant word of the double integer value.
Words 4,5	CPU usage in tens of milliseconds for the last user. Word 4 is the more significant word of the double integer value.

Glossary

capability level

An integer of 0 through 31 that determines a user's access to programs and commands. A superuser has a capability level of 31.

category

A group of functions such as CI commands or FMP routines.

CFT

Category Function Table contains the capability levels for each function.

CIT

Category Index Table contains the address which points to the CFT for all functions in the category.

CRN

Cartridge Reference Number is an integer from 0 through 32767, or two ASCII characters, used to identify a FMGR cartridge.

Default Logon Group

NOGROUP is the default for all system users. Its purpose is to allow systems to operate without setting up groups.

function

An activity carried out by the system on behalf of the user.

group

Several users sharing common resources.

group information

consists of the group identification number, group totals and limits for CPU usage and connect time, group LU access table, number of records in its members list, and list of member records.

LU

Logical Unit is a number used to identify I/O devices.

NOGROUP

The default for all system users. Its purpose is to allow systems to operate without setting up groups.

ORGCPLV

Original Capability Level is the original PROGCPLV given to the program at link time or set with the SECTL utility.

PROGCPLV

Program Capability Level is the limiting factor on what functions a program can perform.

RQUSCPLV

Required User Capability Level is the minimum capability level required to run a program.

security table

Contains the system manager defined set of rules based on capability levels, categories, and functions.

session

The process of logging on, interacting with the system, and logging off.

Session LU Access Table

A 16-word table that provides a means of limiting access to LUs.

UDSP

User-Definable Directory Search Path is a list specifying which directories to search when opening a file, and the order in which they are to be searched.

unique user information

Information associated with an individual user within a group that is used by the system to check the user's limits, initialize the program and directory, and update the user resource information.

USERCPLV

User Capability Level is the capability level of an individual user.

user.group information

Defines associated groups for a user and the resource limitations placed on the user.

VC+

Virtual Code Plus is an HP product that provides multiuser capability to the RTE-A operating system.

Index