
900 Series HP 3000 Computer Systems

MPE/iX System Utilities Reference Manual



HP Part No. 32650-90081
Printed in U.S.A. 1994

Sixth Edition
E0294

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for direct, indirect, special, incidental or consequential damages in connection with the furnishing or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Copyright © 1994 by Hewlett-Packard Company

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. Rights for non-DoD U.S. Government Departments and agencies are as set forth in FAR 52.227-19 (c) (1,2).

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Restricted Rights Legend

Printing History

The following table lists the printings of this document, together with the respective release dates for each edition. The software version indicates the version of the software product at the time this document was issued. Many product releases do not require changes to the document. Therefore, do not expect a one-to-one correspondence between product releases and document editions.

Edition	Date	Software Version
First Edition	November 1987	A.01.00
Update 1	July 1988	A.10.00
Update 2	December 1988	A.20.00
Second Edition	October 1989	A.30.00
Third Edition	April 1990	A.40.00
Fourth Edition	December 1990	B.30.00
Fifth Edition	June 1992	B.40.00
Sixth Edition	April 1994	C.50.00

Preface

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with HP 3000 users, you will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for HP 3000s, not based on PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as *compatibility mode*.

Conventions

UPPERCASE	In a syntax statement, commands and keywords are shown in uppercase characters. The characters must be entered in the order shown; however, you can enter the characters in either uppercase or lowercase. For example: COMMAND can be entered as any of the following: command Command COMMAND It cannot, however, be entered as: comm com_mand comamnd
<i>italics</i>	In a syntax statement or an example, a word in italics represents a parameter or argument that you must replace with the actual value. In the following example, you must replace <i>filename</i> with the name of the file: COMMAND <i>filename</i>
<i>bold italics</i>	In a syntax statement, a word in bold italics represents a parameter that you must replace with the actual value. In the following example, you must replace <i>filename</i> with the name of the file: COMMAND(<i>filename</i>)
punctuation	In a syntax statement, punctuation characters (other than brackets, braces, vertical bars, and ellipses) must be entered exactly as shown. In the following example, the parentheses and colon must be entered: (<i>filename</i>):(<i>filename</i>)
<u>underlining</u>	Within an example that contains interactive dialog, user input and user responses to prompts are indicated by underlining. In the following example, <u>yes</u> is the user's response to the prompt: Do you want to continue? >> <u>yes</u>
{ }	In a syntax statement, braces enclose required elements. When several elements are stacked within braces, you must select one. In the following example, you must select either ON or OFF: COMMAND { ON OFF }
[]	In a syntax statement, brackets enclose optional elements. In the following example, OPTION can be omitted: COMMAND <i>filename</i> [OPTION] When several elements are stacked within brackets, you can select one or none of the elements. In the following example, you can select OPTION or <i>parameter</i> or neither. The elements cannot be repeated. COMMAND <i>filename</i> [OPTION <i>parameter</i>]

Conventions (continued)

[...] In a syntax statement, horizontal ellipses enclosed in brackets indicate that you can repeatedly select the element(s) that appear within the immediately preceding pair of brackets or braces. In the example below, you can select *parameter* zero or more times. Each instance of *parameter* must be preceded by a comma:

[, *parameter*] [...]

In the example below, you only use the comma as a delimiter if *parameter* is repeated; no comma is used before the first occurrence of *parameter*:

[*parameter*] [, ...]

| ... | In a syntax statement, horizontal ellipses enclosed in vertical bars indicate that you can select more than one element within the immediately preceding pair of brackets or braces. However, each particular element can only be selected once. In the following example, you must select **A**, **AB**, **BA**, or **B**. The elements cannot be repeated.

$\left\{ \begin{array}{c} \mathbf{A} \\ \mathbf{B} \end{array} \right\} | \dots |$

... In an example, horizontal or vertical ellipses indicate where portions of an example have been omitted.

Δ In a syntax statement, the space symbol Δ shows a required blank. In the following example, *parameter* and *parameter* must be separated with a blank:

(*parameter*)Δ(*parameter*)

 The symbol  indicates a key on the keyboard. For example,  represents the carriage return key or  represents the shift key.

 *character*  *character* indicates a control character. For example, **Y** means that you press the control key and the Y key simultaneously.

Contents

1. Introduction

2. Utilities Quick Reference

ASOCTBL	2-2
AUTOINST	2-3
BULDACCT	2-4
BUILDINT	2-5
CLKUTIL	2-6
DEBUG	2-7
DIRMIG	2-8
DISCFREE	2-9
DISCUTIL	2-10
DUMP	2-11
EDIT/3000	2-12
FCOPY	2-13
FSCHECK	2-14
GENCAT	2-15
I7DB8CNV	2-16
IOMAP	2-17
KSAMUTIL	2-18
LANGINST	2-19
LINK EDITOR/XL	2-20
LOGTOOL	2-21
MAKECAT	2-22
MKNOD Program	2-23
N7MF8CNV	2-24
NLIOUTIL	2-25
NLUTIL	2-26
NMMGR	2-27
OCA	2-28
OCT	2-29
PATCH	2-30
PXUTIL	2-31
SAINT	2-32
SEGMENTER	2-33
SLPATCH	2-34
SOMPATCH	2-35
SORT-MERGE/XL	2-36
SPIFF	2-37
STANDARDS	2-38
STORE/RESTORE	2-39
SWITCH ASSIST TOOL	2-40

	SYSGEN	2-41
	SYSMAP	2-42
	TERMDSM	2-43
	tic	2-44
	TTUTIL	2-45
	untic	2-46
	V7FF8CNV	2-47
	VERSION	2-48
	VOLUTIL	2-49
3.	ASOCTBL	
	Operation	3-1
4.	BULDACCT	
	Operation	4-1
5.	DISCFREE	
	Operation	5-1
6.	DISCUTIL	
	Operation	6-1
7.	EDIT/3000	
	Operation	7-1
8.	FCOPY	
	Operation	8-2
9.	FSCHECK	
	Operation	9-1
10.	I7DB8CNV	
	Operation	10-1
11.	LANGINST	
	Operation	11-1
12.	LINK EDITOR/XL	
	Operation	12-1
13.	LOGTOOL	
	Operation	13-1

14. MKNOD	
Operation	14-1
15. N7MF8CNV	
Operation	15-1
16. NLIOUTIL	
Operation	16-1
17. NLUTIL	
Operation	17-1
18. PATCH	
Operation	18-1
19. PXUTIL	
Operation	19-1
20. SAINT	
Operation	20-4
21. SLPATCH	
Operation	21-1
22. SOMPATCH	
Operation	22-3
23. SORT-MERGE/XL	
Operation	23-1
24. SPIFF	
Operation	24-1
25. STANDARDS	
The bootstrap	25-2
The initial system loader	25-3
26. SYSMAP	
Operation	26-1
27. tic	
Operation	27-1

28. TERMDSM	
Operation	28-1
29. untic	
Operation	29-1
30. V7FF8CNV	
Operation	30-1
31. VERSION	
Operation	31-1
Index	

Figures

25-1. Boot utility format	25-11
-------------------------------------	-------

Tables

4-1. BULDACCT Processing Options	4-2
7-1. EDIT/3000 Commands	7-2
8-1. FCOPY Functions	8-4
11-1. LANGINST Error Messages	11-10
12-1. Miscellaneous Link Editor/XL Commands	12-2
12-2. RL Link Editor Commands	12-2
12-3. XL Link Editor Commands	12-3
13-1. LOGTOOL Commands	13-3
25-1. Stable storage format	25-3
25-2. Boot or console path	25-4
25-3. Format of flags	25-4
25-4. Nonvolatile memory	25-5
25-5. PDC and boot area for NVM	25-5
25-6. LIF volume label format	25-8
25-7. LIF addressing	25-9
25-8. Main memory layout	25-12
25-9. Console terminal and boot device configuration	25-13
25-10. IODC, IPL, and booted utilities offsets	25-13

Introduction

MPE/iX utilities are programs that provide a dimension of system management and control that ranges from the required or necessary to the helpful or convenient. As such, they help ensure your success as an HP 3000 computer owner by enabling you to fully utilize the potential of your machine.

This manual approaches utilities in two ways: First, chapter 2 provides a quick reference to the basic operation of each utility. This treatment does *not* provide in-depth usage detail. Its intent is to give an overview of each utility and instructions on how to get it started. Second, the remaining chapters are in-depth instructions on how to use utilities that are *not* covered in separate manuals or whose usage, in HP's judgment, is better served by covering both here, *and* in a separate manual. Where a utility is covered both here *and* in a separate manual, the separate coverage is more extensive. The large size and great detail of some utilities prohibit their collective coverage in a single source.

Generally speaking, if you want a quick overview of any particular utility or if you wish to quickly scan all utilities, refer to chapter 2. For example, if you are uncertain that a particular function is covered by a utility you might scan chapter 2. Or, if you know that a utility exists, but need some help remembering what it does or how to start it, you might also refer to chapter 2. For in-depth instructions, see one of the other chapters or the separate reference.

Utilities Quick Reference

This chapter arranges utilities in alphabetical order and provides a quick summary of the basic operation of each one. It does not provide in-depth information.

At the end of each summary, you are told where to find more detailed information. In many cases, you are referred to another chapter in this manual. Where a utility is described in a separate document, however, you are directed toward that source of information.

ASOCTBL

Syntax ASOCTBL

or

RUN ASOCTBL.PUB.SYS

The system responds with > and you should enter:

>devclass=username.acctname

Operation Use ASOCTBL to distribute operator commands for specific devices to standard MPE/iX users.

This utility creates a device class/user table in a file called ASOCIATE.PUB.SYS.

Once a user is included in the association table, he or she gains access to the corresponding device class by the use of the ASSOCIATE command. That user then has exclusive access to the operator commands that control that device until the association is terminated by logging off or issuing the DISASSOCIATE command.

Additional Discussion For more information, refer to Chapter 3 and to the *Controlling System Activity* (32650-90155). For more information on the ASSOCIATE and DISASSOCIATE commands, refer to the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

AUTOINST

Syntax AUTOINST

Operation Use **AUTOINST** to restore the information contained on the FOS and SUBSYS tapes, set up the necessary accounting structure and creates a customized system load tape (SLT).

The fundamental operating system (FOS) and subsystem (SUBSYS) tapes contain programs and utilities for performing specific functions such as compiling programs, copying files, or editing text. You must restore these programs, located on tapes provided with MPE iX, before you can use them.

AUTOINST requires minimal intervention beyond mounting the tapes. It issues messages as it performs its tasks. The messages prompt for any necessary intervention and report on the successful completion of the various procedures.

Additional Discussion For more information, refer to the *HP 3000 MPE/iX Installation and Update Manual* (36123-90001).

BULDACCT

Syntax To use BULDACCT interactively, enter:

```
BULDACCT
```

Or, you may invoke BULDACCT and specify options as part of the ;INFO string by entering:

```
BULDACCT;INFO=" ... "
```

Operation Use BULDACCT to take a *snapshot* of the directory structure on the *source* system and recreate it on a *destination* system. It is especially useful for creating a directory structure on a system without a tape drive where the ;DIRECTORY option of the RESTORE command cannot be used.

BULDACCT also lets you *migrate* a set of accounts from one volume set to another. This is useful with mirror disks, since they do not allow the system volume set to be mirrored, and it may have accounts that you need moved to a different volume set.

BULDACCT will run only on MPE/iX.

Additional Discussion For more information, refer to Chapter 4.

BUILDINT

Syntax RUN BUILDINT.PUB.SYS

or

BUILDINT

Operation Use BUILDINT to build or change Compatibility Mode (CM) intrinsic disk files.

BUILDINT accepts SPL procedure head declarations (OPTION EXTERNAL is required) and optional commands as input data. If no commands are issued, the procedure head declarations are added to the intrinsic file. Any input data that is not a procedure head terminates input. At this point, the program prints a formatted list of all intrinsics and terminates.

Additional Discussion For more information, refer to the *Systems Programming Language Reference Manual* (30000-90024)

CLKUTIL

Syntax From the system console:

```
ISL> CLKUTIL
```

Operation CLKUTIL is a standalone utility that runs only on the physical console at the ISL prompt. It reads and sets the battery-backed-up hardware clock, which should be set to Greenwich Mean Time (GMT). The hardware clock provides the basis for timestamps and time displays that are part of some ISL utilities.

Additional Discussion For more information, refer to the *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042).

DEBUG

Syntax `DEBUG`

or

`RUN PROGNAME;DEBUG`

Operation System programmers use `DEBUG` to set breakpoints within programs and to display and modify data stacks and registers.

To use the `DEBUG` utility, you must have privileged mode (PM) capability. However, the `;DEBUG` option of the `RUN` command is available to anybody and allows users to debug their applications.

Caution Normal MPE safeguards are bypassed in privileged mode. When attempting to modify privileged data on disk, it is possible to destroy file integrity, or the MPE operating system itself. Hewlett-Packard is *not* responsible for changes that you make to the operating system or system files. For more information, talk to your Hewlett-Packard service representative.

Additional Discussion For more information, refer to the *System Debug Reference Manual* (32650-90013) and the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

DIRMIG

Syntax DIRMIG

or

RUN DIRMIG.PUB.SYS

Operation Use the directory migration tool (DIRMIG) to simplify the migration of your environment from MPE V/E-based systems to MPE/iX-based systems.

DIRMIG uses an MPE V/E SYSDUMP tape to transport data to MPE/iX. The data includes the system directory (accounting structure), UDCs, user logging IDs, user files and information specifically related to user volumes.

Additional Discussion For more information refer to the *Migration Process Guide* (32650-90007).

DISCFREE

Syntax

DISCFREE

or

RUN DISCFREE.PUB.SYS;INFO=" [*format*] [, *ldev*] "

or

DISCFREE "[[*format*]][, *ldev*]"

Operation

Use DISCFREE to display information about the system's free disk space in histogram or allocation formats. It determines disk volume fragmentation and transient and permanent disk space limits. DISCFREE also shows total volume space capacity, as well as disk allocation for single volumes or for the whole system.

DISCFREE displays disk allocation data only for mounted MPE/iX volumes, not scratched or unmounted volumes. (You can use the DSTAT command to identify currently mounted volumes.)

Additional Discussion

For more information, refer to Chapter 5 and the following utilities and commands reference sources:

VOLUTIL, *Volume Management Reference Manual* (32650-90045),
DSTAT, *MPE/iX Commands Reference Manual Volumes 1 and 2*
(32650-90003 and 32650-90364).

DISCUTIL

Syntax ISL> DISCUTIL

Operation DISCUTIL is a standalone utility that you can invoke only at the ISL> prompt from the physical console. It lets you request various disk operations when the operating system is not running.

When used in conjunction with the **RECOVER** command of **VOLUTIL**, it can save, and subsequently recover, files from a system that is logically inoperable.

Additional Discussion For more information, refer to Chapter 6 and to the *Volume Management Reference Manual* (32650-90045).

DUMP

Syntax ISL> DUMP

Operation DUMP is a standalone utility that you can invoke only at the ISL> prompt from the physical console. It takes a *snapshot* or dump of system memory at a given point in time, which can help HP support personnel to determine the cause of system problems.

You must precede the DUMP command by a non-destructive boot. During the non-destructive boot sequence, the bootstrap software saves the machine's hardware state at the time that the boot was initiated. DUMP then takes control and dumps the processor internal memory, main memory, and all allocated secondary storage marked for dumping.

Additional Discussion For more information refer to *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042).

EDIT/3000

Syntax EDITOR

Operation Use EDIT/3000 to create and update ASCII files. The Editor commands allow you to insert, delete, replace, modify, search for, and manipulate characters, strings of characters, or entire lines of characters.

You can run EDIT/3000 in batch mode or interactively.

In an interactive session, you enter commands and text through an interactive terminal. Messages and other output (such as prompt characters) from EDIT/3000 are listed on the terminal.

In batch mode, commands and text are supplied through a batch input medium such as a jobstream or magnetic tape. Messages and output are listed on the standard output device, usually a line printer.

Additional Discussion For more information EDIT/3000 refer to Chapter 7 and to the *EDIT/3000 Reference Manual* (03000-90012).

FCOPY

Syntax `FCOPY FROM=filename;TO=filename[:options]`

Operation Use **FCOPY** to copy and translate files. The **FCOPY** command identifies a *from* file, a *to* file, and one or more functions that let you convert data, copy files from other systems, append files, extract subsets of files, display binary files in ASCII format, and other tasks.

- A *from* file is the input file for an **FCOPY** command; it contains the data that you want to copy.
- A *to* file is the output file to which you want to copy the data.

Users with SM capabilities can use **FCOPY** to copy files to MPE accounts outside of their current logon account.

To accommodate the introduction of POSIX in MPE/iX, the **FCOPY** utility lets you copy files from HFS directories into accounts and groups and into other HFS directories. Users with SM capabilities can use **FCOPY** to copy files to MPE accounts outside of their logon account.

Keep in mind the following points when using **FCOPY** to work with HFS files:

- If the *from* file or *to* file name begins with an asterisk (*), the file equation can resolve a filename in HFS syntax that begins with a dot(.) or a slash (/).
- If you are copying files to an HFS directory, you must first use the **FILE** command and specify the file name in HFS syntax
- You cannot use **FCOPY** to copy directories. If the *from* file or *to* file is a directory, you will see an error message.
- You cannot copy compatibility mode (CM) files to HFS directories.

Additional Discussion For more information, refer to Chapter 8 and to the *FCOPY Reference Manual* (32212-90003).

FSCHECK

Syntax FSCHECK

Operation Use the file system check utility ("FCHECK") to detect and repair inconsistencies found in the file directories and file label tables of the MPE/iX operating systems. You also use it to query and display various attributes of these objects. It is a standalone utility and should be the only program running on the system when it is in use.

The FSCHECK utility performs the following functions:

- checks the directory
- checks the label table
- displays the file extent map

The FSCHECK utility also provides consistent MPE/iX subsystem interface, including LISTREDO, REDO, DO, USE, DEBUG, and LOG functionality.

Warning **Do not use this utility without proper service center support. Unauthorized use will void your warranty and may cause data loss!**

Additional Discussion For more information, refer to Chapter 9.

GENCAT

Syntax `RUN GENCAT.PUB.SYS`

or

`GENCAT`

Operation Use `GENCAT` to modify a source catalog or to expand a formatted message catalog (for instance, with messages in the user's native language). It is available to users without any special capabilities.

Additional Discussion For more information, refer to the *Message Catalogs Programmer's Guide* (32650-90021).

I7DB8CNV

Syntax `RUN I7DB8CNV.PUB.SYS`

Operation Use I7DB8CNV to convert the character data in an IMAGE database from any Hewlett-Packard 7-bit national substitution set to ROMAN8. The program is a special version of the program DBLOAD.PUB.SYS, and the conversion is done as part of a database load.

Generally, you run DBUNLOAD.PUB.SYS and DBUTIL.PUB.SYS,ERASE before you run the I7DB8CNV utility.

Additional Discussion For more information, refer to the Chapter 10.

IOMAP

Syntax ISL> IOMAP

Operation IOMAP is a standalone utility that you run at the ISL prompt from the physical console. It identifies the actual I/O configuration of the system and its paths and devices. You can also use **IOMAP's** self test and loopback diagnostics to test I/O system components.

IOMAP displays processor identification (model, identification, processor board revisions, cache sizes, coprocessors, and main memory) and I/O configuration (paths and components for all cards). You can compare this information to the system configuration information to determine the hardware that is physically available versus the hardware that is configured into the system.

Additional Discussion For more information, refer to the *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042).

KSAMUTIL

Syntax To invoke KSAMUTIL, enter:

```
RUN KSAMUTIL.PUB.SYS
```

or

```
KSAMUTIL
```

Operation Use KSAMUTIL to manage *compatibility mode* KSAM files. With KSAMUTIL commands, you can create a **CM KSAM** file, rename both the data and key files, save a temporary file as a permanent file, clear all data from a file, purge a file, and verify the contents and access history of an existing file.

KSAMUTIL runs in either session or batch mode. You can issue MPE/iX commands from within KSAMUTIL by preceding the commands with a colon (:). There are seven KSAMUTIL commands for creating and manipulating CM KSAM files and four commands for displaying file information. The file information may be displayed on the terminal or printed.

Additional Discussion For more information on KSAMUTIL, refer to the *KSAM/3000 Reference Manual* (30000-90079). For information on managing *native mode* KSAM files, refer to *Using KSAM XL* (32650-90168).

LANGINST

Syntax LANGINST

Operation Use **LANGINST** to configure language-specific information onto your HP 3000.

Specifically, **LANGINST** enables you to:

- add a language to the configuration file
- remove a language from the configuration file
- display and modify local formats of a configured language
- display the languages supported by Hewlett-Packard
- display the language currently configured
- modify the system default language

You must log on as **MANAGER.SYS** to run **LANGINST**.

Additional Discussion For more information, refer to Chapter 11.

LINK EDITOR/XL

Syntax LINKEDIT

or

RUN LINKEDIT.PUB.SYS;INFO=*infostring*

Operation Use Link Editor/XL to prepare native mode (NM) compiled object files for execution on 900 Series HP 3000 computers. Link Editor/XL can also create and maintain relocatable and executable libraries.

To invoke Link Editor/XL for interactive use, enter **LINKEDIT** at the MPE/iX prompt. Link Editor/XL then displays its command line prompt (**LinkEd>**) at which you can enter any of the Link Editor/XL commands. To invoke Link Editor/XL and specify an information string, enter a **RUN** command.

Additional Discussion For more information refer *HP Link Editor/XL Reference Manual* (32650-90030).

LOGTOOL

Syntax SYSDIAG
 DUI> RUN LOGTOOL

Operation Use the system and memory log analysis tool (LOGTOOL) to perform a variety of functions on the system log files.

You use LOGTOOL to manipulate two types of log files: system log files and the memory log file. LOGTOOL is available in multi user mode, but you are required to execute at a diagnostic security level for some functions.

Additional Discussion For more information, refer to the *Precision Architecture: HP 3000/9xx & HP 9000/8xx Online Diagnostics Subsystem Utilities Manual* (09740-90021)

MAKECAT

Syntax `RUN MAKECAT.PUB.SYS`

Operation Use MAKECAT to maintain the following message catalogs:

- `CATALOG.PUB.SYS`, which contains system error messages
- `CICAT.PUB.SYS`, which contains the Help catalog
- User-defined catalogs for various applications

Additional Discussion For more information, refer to the *Message Catalogs Programmer's Guide* (32650-90021).

MKNOD Program

Usage `mknod "name c|p major minor [link_name]"`

Operation Use **MKNOD** to create special files in a traditional UNIX format of major and minor numbers. When creating a device link or streams file (character-type file), you must enter both the major and minor numbers. If you enter a major number of 0, **MKNOD** creates a device link file that defaults to the LDEV number specified in the minor number argument. Entering a major number greater than 0 creates a streams file.

Additional Discussion For more information on streams and device link files, refer to the **HPDEVCREATE** intrinsic in the *MPE/iX Ininsics Reference Manual* (32650-90028).

N7MF8CNV

Syntax N7MF8CNV

or

RUN N7MF8CNV.PUB.SYS

Operation Use N7MF8CNV to convert data in EDIT/XL and other MPE text and data files from a Hewlett-Packard 7-bit national substitution character to ROMAN8. N7MF8CNV prompts you for language and a file type of text or data. For each data file, N7MF8CNV prompt you for the starting position and length of each field (portion of a record) to be converted. (For a text file, each record is converted as one field.)

Additional Discussion For more information, refer to Chapter 15 and to the *Native Language Programmer's Guide* (32650-90022).

NLIOUTIL

Syntax NLIOUTIL

or

RUN NLIOUTIL.PUB.SYS;INFO=*infostring*

Operation Use NLIOUTIL to dynamically activate the Native Language I/O (NLIO) subsystem for Asian and Middle East/African (MEA) peripheral devices (terminals and printers). NLIO is the basic input and output system integrated into the operating system for Native Language Support (NLS).

Once activated by NLIOUTIL, properly configured native devices may use the Native Language I/O facility.

Additional Discussion For more information, refer to Chapter 16.

For more information on SYSGEN, refer to the *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042).

For more information on NMMGR, refer to *Configuring Systems for Terminals, Printers, and Other Serial Devices* (32022-90001).

NLUTIL

Syntax

NLUTIL

or

RUN NLUTIL.PUB.SYS

Operation

NLUTIL is a utility program used to verify a variety of Native Language Support (NLS) languages and corresponding character sets available on the operating system. A complete listing may be selected to print on the system printer.

NLUTIL allows the user to display a table showing the currently configured languages and their character set types.

Additional Discussion

For more information, refer to Chapter 17.

A detailed discussion on Native Language Support is contained in the *Native Language Programmer's Guide* (32650-90022).

NMMGR

Syntax NMMGR

Operation Use the Node Management Services Configuration manager to configure your HP 3000's data communications subsystems.

Additional Discussion For more information refer to the following publications *NS3000/XL Network Manager's Reference Manual* (36920-90002), the *NS3000/XL NMMGR Screens Reference Manual* (36922-61003).

OCA

Syntax OCA

Operation Use the object code analyzer to detect migration incompatibilities in compatibility mode applications. You use the output of this tool to formulate a migration plan. When run on MPE/iX systems, it identifies incompatibilities that would prevent the movement of applications from compatibility mode to native mode.

OCA is also available on MPE V/E systems as a component of the migration tool set.

Additional Discussion For more information, refer to the *Migration Process Guide* (30367-90007).

OCT

Syntax

OCTOMP

Operation

Use the object code translator to convert compatibility mode (CM) object code to HP Precision Architecture (PA-RISC) instructions for increased performance. With `OCT`, you can create a new file with translated object code, translate only selected segments of the object code, or add translated segments to another file.

`OCT` translates most CM instructions into HP precision architecture instructions and appends them to the end of the destination file. The resulting file can be executed on either an MPE-V/E based system or an MPE/iX-based system. The effort involved in using `OCT` is less than recompiling but, as in recompiling, you should retest the application to verify proper execution.

Additional Discussion

For more information, refer to the *Migration Process Guide* (32650-90007) and to the *Introduction to MPE XL for MPE V Programmers* (30367-90005).

PATCH

Syntax

PATCH

or

RUN PATCH.PUB.SYS

Operation

Use **PATCH** to access, display, and/or modify a program file's object code without recompiling the program. You can make simple changes to program instructions or to global stack area variables on *compatibility mode* programs.

PATCH requires the memory location of the target program symbols, the beginning locations of each program unit, and the offsets for each line of code from these locations. This data may be gathered in a number of ways depending on the source language of the program.

PATCH supports four subcommands (**D**, **M**, **DG**, and **MG**) to display the code segment contents, to modify the code segment contents, to display the global area of the initial stack, and to modify the global area of the initial stack. **PATCH** can bypass normal MPE/iX safeguards and modify the contents of privileged program files.

Caution

PATCH bypasses normal MPE/iX safeguards and modifies the contents of *privileged* program files. It is, therefore, possible to corrupt system file(s) or the entire operating system. Hewlett-Packard is *not* responsible for modifications that you make to the operating system or system files. For more information, contact your Hewlett-Packard service representative.

Additional Discussion

For more information, refer to Chapter 18.

PXUTIL

Syntax RUN PXUTIL.PUB.SYS

Operation Use PXUTIL to manage the UID/GID databases. The PXUTIL utility requires exclusive access to the databases. This means that any command that needs to modify these files (such as NEWACCT, ALTUSER, and so on) fails during PXUTIL operations. For this reason, no activities that access the databases (NEWACCT, ALTACCT, PURGEACCT, LISTACCT, NEWUSER, ALTUSER, PURGEUSER, LISTUSER, and logon commands) should be attempted on the system while PXUTIL. You must have SM capability to run PXUTIL.

Additional Discussion For more information, refer to Chapter 18.

SAINT

SAINT is a standalone initialization utility that you use to analyze system libraries (that contain system object modules, or SOMs) to produce *bootable images*. A bootable image is a file that can be copied directly to memory and executed without modification.

SAINT primarily depends on the system library file for input. The format of the library file is defined in the *SOM Architecture Control Document*.

Warning

Do not use this utility without service center support. Unauthorized use will void your warranty and may cause data loss.

Additional Discussion

For more information, refer to Chapter 20.

SEGMENTER

Syntax `SEGMENTER`

Operation Use `SEGMENTER` to manage and prepare compatibility mode code segments.

Invoked directly with the `SEGMENTER` command, `SEGMENTER` lets you manage code segments in USLs (user subprogram libraries), RLs (relocatable libraries) and SLs (segmented libraries), including the ability to group RBMs (relocatable binary modules) into code segments. Invoked indirectly (at prep time) `SEGMENTER` lets you define run-time parameters (using `PREP` parameters) and to group compatibility mode program statements into RBMs and code segments (using source program statements).

Additional Discussion For more information, refer to the *MPE Segmenter Reference Manual* (30000-90011) and to the *HP Link Editor/XL Reference Manual* (32650-90030).

SLPATCH

Syntax SLPATCH

or

 RUN SLPATCH.PUB.SYS

Operation Use SLPATCH to display or modify the contents of a segmented library (SL) file.

Caution SLPATCH bypasses normal MPE/iX safeguards and will modify the contents of *privileged* SLs. It is, therefore, possible to corrupt the SL or the entire operating system. Hewlett-Packard is *not* responsible for modifications that you make to the operating system or system files. For more information, contact your Hewlett-Packard service representative.

Before using this utility, you should be familiar with machine-executable instructions and the internal format of segmented library files in the HP 3000 system environment.

Additional Discussion For more information, refer to Chapter 21, and to the *MPE Segmenter Reference Manual*(30000-90011).

SOMPATCH

Use SOMPATCH for binary modification (or *patching*) of a native mode spectrum object module (SOM) program or library file.

Warning

Do not use this utility without service center support. Unauthorized use will void your warranty and may cause data loss.

SORT-MERGE/XL

Syntax	SORT	MERGE
	OR	OR
	RUN SORT.PUB.SYS	RUN MERGE.PUB.SYS
Operation	Use SORT to sort files based on single-key or multiple-key items. Use MERGE to merge data from two or more sorted files into a single, new file.	
	SORT-MERGE/XL operates as a standalone utility (either interactively or in batch mode), or from within a program. You can use SORT-MERGE/XL to sort or merge data in various ways. Some sequences that you might choose as the basis for sorting or merging data could be ■ alphabetically in either an ascending or descending order ■ numerically in either an ascending or descending order ■ alphabetically or numerically based on a single key data items ■ alphabetically or numerically based on multiple key data items ■ define a unique collating sequence for your application ■ merge two or more sorted files into a new merged file	
Additional Discussion	For more information, refer to the Chapter 23 in this book and to <i>SORT-MERGE/XL General User's Guide</i> (32650-90082).	

SPIFF

Syntax SPIFF

Operation Use the Native Mode Spooler Interface Facility to list, manipulate, and transfer spooled device files (spoolfiles) that are created and maintained by MPE/iX. **SPIFF** is the MPE/iX replacement for the MPE compatability mode **SP00K5** program.

Additional Discussion The *Native Mode Spooler Reference Manual* (32650-90166) presents a detailed description of the **SPIFF** utility and its commands.

STANDARDS

The system bootstrap, initial program load (IPL) and initial system load (ISL) standard provides a standard interface through which any Hewlett-Packard Precision Architecture (PA-RISC) computer can boot any operating system. The standard also provides a common user interface for booting PA-RISC systems.

Warning

The use of this information without service center support will void your warranty and may cause data loss.

Additional Discussion

For more information, refer to Chapter 25.

STORE/RESTORE

Syntax To invoke STORE enter:

```
STORE filesset[; parameters]
```

To invoke RESTORE enter:

```
RESTORE storfile[; parameters]
```

Operation

Use STORE/RESTORE to store and restore one or more files and directories to and from tape. It has special options that allow you to store files for backup, transport, or archival purposes.

For example, to store all files in all groups in the MFG account, enter:

```
FILE T;DEV=TAPE  
STORE @. @.MFG;*T;SHOW
```

The system issues a file equation with your logon user name as the *formal file designator*. You can implicitly use that file equation by omitting a file reference in your STORE command. For example:

```
STORE @. @.MFG;;SHOW
```

To restore all files in all groups in the MFG account, enter:

```
FILE T;DEV=TAPE  
RESTORE *T;@. @.MFG;SHOW
```

You may implicitly reference the system-generated file equation by omitting the *T. You must retain the semicolon as a placeholder, however.

Additional Discussion

For more information, refer to *Performing System Manager Tasks* (32650-90004) and *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

SWITCH ASSIST TOOL

SYNTAX

SWAT

or

RUN SWAT.PUB.SYS

Operation

Use the switch assist tool (SWAT) to simplify the process of implementing an application that has modules written both in native and compatibility modes.

SWAT takes input from the user and generates output in the form of PASCAL/XL source code.

Additional Discussion

For more information, refer to *Switch Programming Guide* (32650-90014).

SYSGEN

Syntax SYSGEN

- Operation** Use **SYSGEN** to modify your system configuration. It consists of a global module and four configurator modules:
- The IO configurator, for configuring local devices
 - The LOG configurator, for configuring user and system logging processes
 - The MISC configurator, for configuring limits on system resources, jobs and sessions
 - The SYSFILE configurator, for changing system libraries, programs and message catalogs

The changes you make with SYSGEN are written to disk or to tape, and only take effect when you restart the system.

Additional Discussion For more information, refer to *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042) and to *Performing System Manager Tasks* (32650-90004).

SYSMAP

Syntax

```
SYSDIAG  
DUI> RUN SYSMAP  
ENTER MAP>
```

Operation Use **SYSMAP** to display the hardware configuration of a system. It displays the device type, product number, logical device number (LDEV) and device address of I/O devices, and the CPU and memory boards.

SYSMAP is part of the online diagnostics subsystem. To use **SYSMAP**, you start the diagnostic subsystem by entering **SYSDIAG** and, at the **DUI>** prompt, enter the command **RUN SYSMAP**. You then choose one of three basic map options:

- **CPUMAP**, to get information about the cpu
- **MEMMAP**, to get information about the size, location and other information about system memory
- **IOMAP**, to get general I/O configuration information, information about a specific device class, or information about devices on or below a specific physical address

Additional Discussion For more information, refer to Chapter 26 and to the *Precision Architecture: HP 3000/9xx & HP 9000/8xx Online Diagnostics Subsystem Utilities Manual* (09740-64007).

TERMDSM

Syntax SYSDIAG
 DUI > RUN TERMDSM

Operation Use TERMDSM to diagnose, dump, and reset logical devices, ports, and data communications and terminal controllers (DTCs). You also use it to check the status of ports and DTCs.

Additional Discussion For more information, refer to *Troubleshooting Terminal, Printer, and serial Device Connections* (32022-61002).

tic

Syntax `tic.hpbinsys [-v[n]] -c file`

Operation Use the `tic` utility to compile source `terminfo` descriptions.

`tic` installs the compiled entry under the `/usr/lib/terminfo` directory hierarchy, unless you set the `TERMINFO` environment variable, in which case it places results in the directory it points to instead. Entries are stored in directories that match the first character of their name. The entry for the VT-100 terminal, for example, is stored in `/usr/lib/terminfo/v/vt100`.

Additional Discussion For more information, refer to Chapter 27.

TTUTIL

Syntax `RUN TTUTIL.PUB.SYS`

or

`TTUTIL`

Operation Use `TTUTIL` to create, view and modify an existing terminal or printer type file. You can modify characteristics of serial port connections such as flow control, modem control, printer control and character handling. To do so, you enter the terminal or printer type file name and then specify a function.

Additional Discussion For more information on `TTUTIL`, refer to the *Customizing Terminal and Printer Type Files with Workstation Configurator* (5959-2870).

untic

Syntax `untic.hpbin.sys [ term ]`

Operation Use `untic` to decompile a `terminfo` binary file into its source format. If a `TERMINFO` environment variable is set, the `untic` utility searches the specified directory; otherwise, `untic` assumes the file is in the directory `/usr/lib/terminfo`. The output of an `untic` decompile is sent to the standard output.

Additional Discussion For more information, refer to Chapter 29.

V7FF8CNV

Syntax V7FF8CNV

or

RUN V7FF8CNV.PUB.SYS

Operation Use V7FF8CNV to convert text and literals in VPLUS/XL forms files from a Hewlett-Packard 7-bit national substitution character set to ROMAN8.

Additional Discussion For more information, refer to Chapter 30.

VERSION

Syntax

VERSION

or

VERSION *filename*

or

VERSION "*filename* [,*search string*]"

Where

file(s) *File(s)* is the name of a program file or a wild carded file set.

search string *Search string* is the name of a particular \$version string in a system object module (SOM)(Not applicable for CM program files.) Quotes are required if a search string is specified.

Operation

Use **VERSION** to display information about compatibility mode and native mode files.

- For compatibility mode (CM) program files, **VERSION** displays segment, stack, data reference base, and capability information.
- For native mode (NM) executable files, **VERSION** displays information on procedures, libraries, capabilities, stack, heap, entry names, and \$version strings.
- For NM object files and nonexecutable library files, **VERSION** displays \$version strings.

VERSION expects a file name or a wild carded file set as input. If you invoke **VERSION** without entering a file name or a file set, the **VERSION>** prompt appears. **VERSION** continues to prompt you for input until you enter **EXIT** or a colon (:).

If the input is a file set, **VERSION** processes every file in the set even if an error occurs during processing. If there is an error opening a file, you will see the file system error in addition to the **VERSION** error message.

Additional Discussion

For more information, refer to Chapter 31. For more information on the \$version strings found in the SOM, refer to the *HP Pascal Programmer's Guide* (31502-90002).

VOLUTIL

Syntax

```
VOLUTIL  
volutil> command name
```

or

```
RUN VOLUTIL.PUB.SYS  
volutil> command name
```

Operation Use VOLUTIL to manage and maintain individual volumes, volume sets, and volume classes and to make inquiries about their contents, availability, and status.

VOLUTIL commands are organized into four groups, based on the object they manipulate. All commands that control sets end with SET; those that control classes end with CLASS; the commands that control volumes end with VOL; and the last group consists of miscellaneous commands.

You can use any MPE/iX system command from within VOLUTIL by first entering a colon (:), then the command.

Additional Discussion For more information, refer to *Volume Management Reference Manual* (32650-90045).

ASOCTBL

The ASOCTBL utility lets users with SM or OP capability distribute *operator* commands for specific devices to other system users. It creates an association table in `ASSOCIATE.PUB.SYS`, where the users are identified with the device classes that they may associate.

Once a user is included in the association table, he or she can use the `ASSOCIATE` command to gain access to the corresponding device class. While a device class is under a user's control, status messages for the devices in that class appear on the user's `$STDLIST` device, and not on the system console. That user continues to have exclusive access to the operator commands that control that device until the user terminates the association by logging off or by issuing the `DISASSOCIATE` command.

Operation

To invoke ASOCTBL enter:

ASOCTBL

or

RUN ASOCTBL.PUB.SYS

The system responds with a `>` prompt and you should enter the following:

`> devclass=username.acctname`

Parameters

<i>devclass</i>	The name of a logical device class configured with <code>SYSGEN</code> or <code>MMMGR</code> . When you specify a device class, all devices assigned to that class are affected by the <code>ASSOCIATE</code> command.
<i>username</i>	The name assigned to the user by the Account Manager. It must contain from one to eight alphanumeric characters, beginning with an alphabetic character. The <i>username</i> parameter may be replaced by <code>@</code> to indicate all users. For example, <code>@.acctname</code> enables all users in the specified account.

acctname The name created by the System Manager to identify the account. It must contain from one to eight alphanumeric characters, beginning with an alphabetic character. The *acctname* parameter may be replaced by @ to indicate all accounts. For example, *username.@* enables all users of the specified name in any account.

@.@ Enables all users in all accounts.

ASOCTBL expects input from an ASCII file or a terminal. If data is read from a file, the formal file designator is *INPUT*. ASOCTBL reads the input file until it finds an end-of-file (EOF) indicator or a statement beginning with **EXIT** in column 1. When ASOCTBL finds errors in the *INPUT* file, it scans remaining data and then terminates without updating *ASOCIATE.PUB.SYS*.

If a file equation for *INPUT* does not exist, ASOCTBL prompts for input from the terminal.

You may not use a text editor to directly modify the *ASOCIATE* file.

Using ASOCTBL with an ASCII File

To run ASOCTBL with an ASCII file you must first create the file using a text editor such as EDIT3000. For example, in the following ASCII file the users and accounts, *MGR.MNFG*, *USER.UTILITY* and *JACK.MR KTG* will be able to associate with *LP*, *TAPE* and *LP2* respectively.

```

EDITOR
/ADD
1 LP = MGR.MNFG
2 TAPE = USER.UTILITY
3 LP2 = JACK.MRKTG
4 EXIT
5 //
/KEEP ASOTFILE,UNN
/E

```

Once you have created the file and named it, you must use it in a file equation and then invoke the file. To continue the preceding example, you would enter:

```

FILE INPUT=ASOTFILE
ASOCTBL

```

ASOCTBL displays the devices and the users specified in the ASCII file *ASOTFILE*.

Using ASOCTBL Interactively

To use ASOCTBL interactively enter:

```

ASOCTBL

```

At the ASOCTBL prompt, enter devices and the user names you want to associate. For example:

```
> LP = MGR.MNFG
> TAPE = USER.UTILITY
> LP2 = JACK.MRKTG
```

To terminate ASOCTBL, enter:

```
> EXIT
```

Using Wildcards in User and Account Names

You may use wildcards to specify an association for a group of users, such as those using the same account. For example, the command below associates all users logged onto the **FINANCE** account, and the user **JACK** logged onto *any* account with the device class **LP**:

```
ASOCTBL
> LP = @.FINANCE,JACK.@
> EXIT
```

In this example user **JACK.MRKTG**, all users logged onto the **FINANCE** account, and the user **JACK** logged onto *any* account may associate with device class **LP**.

Removing Entries from ASOCIATE.PUB.SYS

ASOCTBL builds a new version of **ASOCIATE.PUB.SYS** each time it is invoked and successfully terminates. Therefore, to remove entries invoke ASOCTBL and specify only the the entries you currently want.

Listing the Association Table

To list the association table in **ASOCIATE.PUB.SYS** (users and the devices to which they may associate) enter:

```
RUN ASOCTBL.PUB.SYS,LIST
```

Fatal Errors

The following error messages indicate a condition where no modifications are made to the association table. Although you may continue to make inputs to the system without damage, you should exit from ASOCTBL and begin again.

UNABLE TO DELETE OLD 'ASOCIATE.PUB.SYS' FILE

EXPECTED AT LEAST 3 PARAMETERS, LDEV = USER.ACCT

= MUST FOLLOW LDEV

UNABLE TO OPEN INPUT FILE

CLASS NAMES ARE LIMITED TO 8 CHARACTERS

NO SUCH CLASS IN THIS SYSTEM

EXPECTED . FOLLOWING USER NAME

UNABLE TO OPEN NEW ASOCIATE.PUB.SYS FILE

EXPECTED , FOLLOWING EACH USER.ACCT

USER AND ACCOUNT NAMES ARE
1 TO 8 ALPHANUMERIC CHARACTERS
OR "@".

Additional Discussion

For more information refer to *Controlling System Activity* (32650-90155). For more information on the ASSOCIATE and DISASSOCIATE commands refer to *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

BULDACCT

The BULDACCT utility lets users with SM capability take a *snapshot* of the directory structure on a *source* system running MPE/iX and recreate it on a different *destination* system. It is especially useful for creating a directory structure on a system without a tape drive where the ;DIRECTORY option of the RESTORE command cannot be used.

BULDACCT also lets you *migrate* a set of accounts from one volume set to another. This is useful with mirror disks, since they do not allow the system volume set to be mirrored, and it may have accounts that you need moved to a different volume set.

BULDACCT has been enhanced to support the hierarchical file system of MPE/iX. As the default, BULDACCT automatically saves all accounts, groups, users, hierarchical directories and the ACDs associated with the directories. However, if BULDACCT detects that the operating system does not support POSIX, it does not look for any hierarchical directories.

In addition, three new options have been added to BULDACCT: %NODIRS%, %ROOT% and %NOROOT%. These options let you control the migration of root and non-root hierarchical directories when using BULDACCT. Each option is explained in the “Operation” section.

Operation

To invoke BULDACCT and use the utility interactively, enter:

```
BULDACCT
```

The utility responds with the BULDACCT: prompt, where you enter options from the list shown in Table 4-1. You may use a maximum of 80 characters in the command line.

Or, you may invoke BULDACCT and enter options when you issue the command, like this:

```
BULDACCT;INFO=" ... "
```

In this format, you insert options from the list shown in Table 4-1 in the ;INFO string. You may use a maximum of 240 characters.

The processing options for the BULDACCT utility are shown in Table 4-1. The capital letters denote keywords and the lower case letters denote user selected names.

Table 4-1. BULDACCT Processing Options

Option	Description
%HELP	displays detailed information about the options
%QUIT	quits the interactive prompt
[acct_list]%NODIRS	prohibits migration of any hierarchical directories to BULDJOB1
[acct_list]%ROOT	migrates hierarchical directories immediately under ROOT ("/")
[acct_list]%NOROOT	prohibits migration of the ROOT hierarchical directories to BULDJOB1
acct_list%VSACCT=user_set	migrates accounts and groups to the user_set volume set
acct_list%VS=user_set	migrates only the groups to the user_set volume set
acct_list%UV[=user_set]	selects accounts with at least one group on any non-system volume set
acct_list%FROMVS=home_volume_set_name	specifies account list with the home volume set as a selection criteria

The BULDACCT utility produces two job files: BULDJOB1 and BULDJOB2, which you use in the following way:

- To create users, hierarchical directories, groups and accounts on the destination system, move BULDJOB1 to that system and stream it.
- To create user-defined command files (UDCs) on the destination system, first move all of the files using the **STORE** and **RESTORE** commands (or, if the destination system is on a network, you can directly copy the UDC files.) Then, to set system, account and user level UDC's, move BULDJOB2 to the destination system and stream it.

BULDACCT always creates two files, BULDJOB1 and BULDJOB2, even if only one is needed.

Caution

If the *destination* system already contains user, group or account names that existed on the *source* system, BULDJOB1 will change their attributes to match those of the *source* system. If the *destination* system already contains hierarchical directories that existed on the *source* system, BULDJOB1 will change the ACDs to match the source system.

Recreating the complete directory structure

To recreate the entire directory structure of the source system and set all system, account, and user level UDCs, enter:

```
BULDACCT
```

To create the new directory on the destination system enter:

```
STREAM BULDJOB1
```

If you wish to *reset* UDC's, use the :RESTORE command to restore UDC files and then enter:

```
STREAM BULDJOB2
```

Recreating selected accounts on a new system

You may specify a selected list of accounts in the accounts list portion of the ;INFO string. For example, to select all those accounts beginning with S and all those accounts ending with P enter:

```
RUN BULDACCT;INFO="S@,@P"
```

Then to create the new directory entries on the destination system, enter:

```
STREAM BULDJOB1
```

Migrating root and hierarchical directories

When BULDACCT is processing individual accounts or a subset of "all accounts" ("@"), it does *not* process any hierarchical directories directly under ROOT ("/"). It only processes the hierarchical directories within each of the specified accounts. For example, the following command processes all groups, users and hierarchical directories in accounts GYPSY1 and GYPSY2:

```
RUN BULDACCT; INFO="GYPSY1,GYPSY2"
```

To prevent BULDACCT from picking up any hierarchical directories, use the %NODIRS option. To prevent BULDACCT from picking up any hierarchical directories under ROOT ("/"), use the %NOROOT option. If you use both ROOT and %NOROOT, %NOROOT has precedence.

Migrating a specific account to a non-system volume set

To migrate a selected set of accounts to a non-system volume set use the VSACCT option of BULDACCT. In this example the source computer and destination computer are the same. The basic BULDACCT syntax is:

```
RUN BULDACCT;INFO="[acct_list]%VSACCT=user_vol_set"
```

In this example `acct_list` is the list of accounts to be migrated. The default account list is @ or *all* accounts; `user_vol_set` must be a valid volume set name.

The following is a sample of the exact steps you would use to migrate an account called **GYPSY** to a non-system volume set called **target_vol_set**.

1. Log on as **manager.sys** and store all the files in the selected account:

```
STORE @.@.GYPSY;;SHOW
```

2. Run **BULDACCT** with the **VSACCT** option to create job files **BULDJOB1** and **BULDJOB2**:

```
RUN BULDACCT;INFO="GYPSY%VSACCT=target_vol_set"
```

3. Close system level UDC's residing in the **GYPSY** account (if any):

```
SETCATALOG udc_file;SYSTEM;DELETE
```

4. Remove the account from its existing location and make sure there is not a **GYPSY** account on your **target_vol_set**:

```
PURGEACCT GYPSY  
PURGEACCT GYPSY;ONVS=target_vol_set
```

5. Rebuild the account, group and user directory structure for **target_vol_set**:

```
STREAM BULDJOB1
```

6. Restore all **GYPSY** account files, making sure that the files whose creators have been purged are also restored:

```
RESTORE ;@.@@;SHOW;VOLSET=target_vol_set;CREATE=CREATOR
```

(You may wish to add **;OLDDATE** to this command to preserve the *modify*, *access* and *create* dates.)

7. Reset all user and account level UDC's:

```
STREAM BULDJOB2
```

8. If a system level UDC was uncataloged in step 3 then set it again:

```
SETCATALOG udc_file;SYSTEM;APPEND
```

Migrating groups to a non-system volume set

If a given account already exists on a non-system volume set, you may use the **VS** option to migrate groups of that account to that volume set. The basic **BULDACCT** syntax is:

```
RUN BULDACCT;INFO="[acct_list]%VS=user_vol_set"
```

The following is an example of the exact steps you would use to migrate the groups in the **GYPSY** account from one volume set (system or non-system) to a user (non-system) volume set called **target_vol_set**:

1. Logon as manager.sys and store all the files in the **GYPSY** account:

STORE @.@.GYPSY;;SHOW

2. Issue the **BULDACCT** command with the **VS** option:

RUN BULDACCT;INFO="GYPSY%VS=target_vol_set"

3. Unset system level UDC's in the **GYPSY** account:

SETCATALOG udc_file;SYSTEM;DELETE

4. Remove the **GYPSY** account from the system volume set:

PURGEACCT GYPSY

5. Rebuild the directory structure for **target_vol_set**:

STREAM BULDJOB1

6. Restore files to the **GYPSY** account; making sure that the files whose creators have been purged are also restored.

RESTORE ;@.@.@;SHOW;VOLSET=target_vol_set;CREATE=CREATOR

(You may wish to add ;OLDDATE to this command to preserve the *modify*, *access* and *create* dates.)

7. Reset all user/account level UDC's:

STREAM BULDJOB2

8. If a system level UDC was unset in step 3 reset it:

SETCATALOG udc_file;SYSTEM;APPEND

Qualifying the account list

You can use the **FROMVS** option of the **BULDACCT** utility to specify the account list on particular volume sets. For example, to dump the directory information of all the listed accounts that have at least one group on **home_volume_set_name**, enter:

```
:RUN BUILDACCT: INFO="account_list %FROMVS=home_volume_set_name"
```

A minimum of one group linkage is required since the build account program does the scanning on the system volume set. To select all accounts on the system that have a group on some volume set, you can substitute a wild card (@) for part or all of the account list.

The complete information to rebuild the accounts, groups and users on the system, as well as the specified volume set, is dumped in the job scripts. If you want to migrate these accounts to a different volume set, use the **VSACCT** option in conjunction with the **FROMVS** option. For example, to select all the accounts which have at least one group with **HOMEVS=home_volset** and dump information to rebuild these accounts with *all* their groups on the target volset, enter:

```
:RUN BULDACCT: INFO="@ %FROMVS=home_volset %VSACCT=target_volset"
```

You may use the **UV** option to select only those accounts which have at least one group on any user (non-system) volume set. For example:

```
RUN BULDACCT;INFO="[acct_list]%UV=user_vol_set"
```

=user_vol_set is optional. If omitted, **BULDJOB1** will create the directory structure in the *system* domain of the *destination* system. If **user_vol_set** is included, **BULDJOB1** will create the directory structure in the *non-system* domain of the *destination* system. In either case, **BULDACCT** selects only those accounts on the source system which have a group on a user (non-system) volume set.

For example, in the following command, the account **GYPSY1** has groups on a user volume, whereas **GYPSY2** does not. As a result, when you stream **BULDJOB1**, it will create the directory only for **GYPSY1**.

```
RUN BULDACCT;INFO="GYPSY1,GYPSY2%UV=user_vol_set"
```

If, in the preceding example, you omit **=user_vol_set**, **BULDJOB1** would create the directory structure for **GYPSY1** in the *system* domain. Since **=user_vol_set** is included, **BULDJOB1** creates the directory structure for **GYPSY1** in the *non-system* domain on volume set **user_vol_set**.

Getting help

To get information about the **BULDACCT** utility, you can:

- Issue the **BULDACCT** command (without the **;INFO** string) and, at the prompt, enter **%HELP**
- Enter **%HELP** in the **;INFO** string as follows:

```
RUN BULDACCT;INFO="%HELP"
```

DISCFREE

Use the DISCFREE utility to display information about disk volume fragmentation and the allocation of transient and permanent space on the disk. DISCFREE also shows total volume space capacity, as well as disk allocation for single volumes or for the whole system.

Transient space refers to temporary, fluctuating volume space. Permanent space is volume space set aside for job/session temporary files, permanent disk files, and operating system directories and other data.

DISCFREE displays disk allocation data only for mounted MPE/iX volumes, and not for unmounted or scratched volumes. (If necessary, use the DSTAT command to identify currently mounted volumes.)

Operation

To invoke DISCFREE enter:

DISCFREE

or

RUN DISCFREE.PUB.SYS; INFO=" [format] [, ldev] "

or

DISCFREE " [format] [, ldev] "

where

format is the specified display type. The *format* default is **A**.

ldev Specifies a particular system volume. The *ldev* default is no *ldev*, causing all volumes to be displayed.

If you do not specify either of the parameters, DISCFREE prompts for them. If you enter *format* but not *ldev*, DISCFREE reports on all logical disk devices. To report on a particular volume, enter the name of the volume set. For example, to display information about the volume USER_VOL_SET, you would enter:

RUN DISCFREE.PUB.SYS; INFO=" , ,USER_VOL_SET"

The format values you can specify are:

- | | |
|---|--|
| A | A <i>histogram</i> showing numbers of blocks by size categories. |
| B | Shows transient and free space allocations in sectors. |
| C | Shows transient and free space allocations in percentage of total device capacity. |
| D | Shows disk allocation summary in sectors. |
| E | Shows disk allocation summary in percentage of total space. |

Displaying the histogram

To display a *histogram* for *ldev* 1 only enter:

```
DISCFREE "A,1"
```

To display the histogram for *all* disk devices enter:

```
DISCFREE "A"
```

Or, you may enter the DISCFREE command (without any options) and, at the prompt, enter only the format and ldev or only the format. For example:

```
Enter [<format>][,<ldev>] : "A,1"
```

```
Enter [<format>][,<ldev>] : "A"
```

Displaying the allocation summary

To display the *allocation* summary as a percent of total space on all logical disk devices, enter:

```
DISCFREE "C"
```

If you omit quotation marks, the default is the specified format for *all* devices.

DISCFREE displays the following information:

DEVICE SIZE	The capacity, in sectors, of the device.
TRANS SPACE	The number of sectors currently allocated for transient space. Transient space is used for objects which are not permanent files, such as stacks, heaps and system tables.
MAX TRANS SPACE	An upper limit on the number of sectors that may be allocated for transient space. TRANS SPACE should <i>not</i> exceed this number although it <i>may</i> if the percentage of MAX TRANS SPACE space is lowered using VOLUTIL.
FREE SPACE	The amount of space on a device currently <i>not</i> used for transient or permanent space. Space available for use.
AVAIL TO TRANS SPACE	The amount of space on the device currently available for use as transient space. The difference between MAX TRANS SPACE and TRANS SPACE. If this difference exceeds the amount of FREE SPACE, DISCFREE displays the amount of FREE SPACE.
PERM SPACE	The number of sectors currently allocated for permanent space. User and system files use permanent space.
MAX PERM SPACE	An upper limit on the number of sectors that may be allocated for permanent space. PERM SPACE should <i>not</i> exceed this number although it <i>may</i> if the percentage of MAX PERM SPACE space is lowered using VOLUTIL, or if enough space is consumed by files brought in during an UPDATE.
AVAIL TO PERM SPACE	The amount of space on the device currently available for use as permanent space. The difference between MAX PERM SPACE and PERM SPACE.

AVAIL TO TRANS SPACE and AVAIL TO PERM SPACE may be *negative* values. For example, if TRANS SPACE exceeded MAX TRANS SPACE, AVAIL TO TRANS SPACE would be negative. This could occur if VOLUTIL were used to change the maximum percentage of MAX TRANS SPACE to a level lower than currently used for TRANS SPACE. Also, if enough permanent space is used by files brought in during an UPDATE, PERM SPACE may exceed MAX PERM SPACE, which would make AVAIL TO PERM SPACE a negative number.

Additional Discussion

For more information on the general topic of disk volumes refer to *Volume Management Reference Manual* (32650-90045). For more information on the DSTAT command refer to *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

DISCUTIL

DISCUTIL is a *standalone* utility that lets you perform various disk operations without the operating system. It is particularly important when you need to move files to tape from disks on a system that is logically inoperable, such as one with directory damage. (Once you have done so, you then use the **RECOVER** option of the **VOLUTIL** utility on this tape to move the files back onto disk.)

You start the utility by entering **DISCUTIL** at the Initial System Load (ISL) prompt from the system console. (You may not use **DISCUTIL** at another terminal.) Once **DISCUTIL** is loaded and running, the operating system is *not* available.

Operation

You may invoke **DISCUTIL** only from the **ISL>** prompt. To display the **ISL>** prompt and start the utility, perform these steps:

1. Initiate a soft reset of the system by entering **CTRL B**.
2. At the **CM>** prompt, enter the following command to start the autoboot process:

 RS **ENTER**
3. When you see the message **TO OVERRIDE, PRESS ANY KEY WITHIN 10 SECONDS**, press **RTN** immediately.
4. When you see the message **Interact with IPL (Y or N)?**, enter **Y**. The system displays the **ISL>** prompt.
5. To start **DISCUTIL**, enter:

ISL> **DISCUTIL**

The **DISCUTIL** banner will appear followed by the **discutil>** prompt.

Note

To see other utilities available *before* the system is started enter **LS** at the **ISL>** prompt.

DISCUTIL Commands

At the `discutil>` prompt, you may enter any of the following commands:

CONFIGURE	Configures additional devices. (The system console, all disks that were mounted at the time of the failure, and the tape drive are configured by default.)
DISMOUNT	Dismounts a disk volume making it inaccessible to DISCUTIL .
DO	Reexecutes a command in the command history stack.
DSTAT	Displays information about each mounted volume.
EXIT	Terminates DISCUTIL and reboots the system.
HELP	Lists available commands.
LISTREDO	Displays the command history stack.
MOUNT	Mounts a disk volume making it accessible to DISCUTIL .
PDEV	Lists the disks that are currently configured and mounted.
REDO	Modifies and reexecutes a command from the command history stack.
SAVE	Saves user-specified files from disks to tape.
SHOWDEV	Displays information about each configured device.
TAPE	Displays and selects the current tape <i>ldev</i> .
UNCONFIG	Unconfigures currently configured devices.

The remainder of this chapter describes each of the commands. Remember that these commands, though they may appear similar to MPE/iX commands or the commands used with other utilities, operate only as described here.

CONFIGURE

The **CONFIGURE** command is used to dynamically add or configure new/additional devices into the current configuration. You may access only configured devices with **DISCUTIL**.

Note

By default, **DISCUTIL** configures the system console, a tape drive, and all disks indicated as being mounted in **MPE XL** tables. For this reason, you need not use the **CONFIGURE** command on any of these devices.

To invoke the **CONFIGURE** command enter:

```
discutil>CONFIGURE [LDEV=] ldev  
                  [CLASS=] devclass [PATH=] path
```

For example:

```
discutil>CONFIGURE 21 DISC 6/4.0.0
```

DISMOUNT

Use the **DISMOUNT** command to logically dismount a volume. This command removes the volume entry corresponding to the specified *ldev* from **DISCUTIL**'s mounted volume table, thus making it inaccessible to **DISCUTIL**. After a volume has been logically dismounted, the media or disk pack on the corresponding *ldev* can be spun down, new media mounted and spun up, then logically mounted with the **MOUNT** command. **DISCUTIL** will then recognize the new media.

To invoke the **DISMOUNT** command enter:

```
discutil>DISMOUNT [LDEV=] ldev
```

For example:

```
discutil>DISMOUNT 4
```

DO

Use the **DO** command to re-execute a command from the command history stack/queue.

To invoke **DO** enter:

```
discutil> DO [ [CMD=] cmdid]
```

For example, to re-execute the fifth command, enter:

```
discutil>DO 5
```

DSTAT

Use this command to display information about each mounted volume (i.e., any volume that is currently listed in the mounted volume table). To invoke DSTAT enter:

```
discutil>DSTAT
```

DSTAT displays the *ldev* on which the volume is mounted, the type ID of the device, the type (STATUS) of the volume, the volume name, and the physical path as shown in the example below:

LDEV-TYPE	STATUS	VOLUME (VOLUME SET-GEN)	PATH
-----	-----	-----	-----
1-079350	MASTER	MPEXL_SYSTEM_VOLUME_SET:MEMBER1	8.0.0
2-079350	MEMBER	MPEXL_SYSTEM_VOLUME_SET:MEMBER2	8.0.1
3-079350	MEMBER	MPEXL_SYSTEM_VOLUME_SET:MEMBER3	8.0.2
14-079350	MASTER	OFFICIAL:MASTER	8.0.4

EXIT

Use the EXIT command to terminate DISCUTIL and automatically begin a system reboot. To exit DISCUTIL enter:

```
discutil>EXIT
```

When the ISL> prompt reappears you may use the START command to restart your system or choose another *standalone* utility. (To see a list of the utilities available to you, enter LS.)

HELP

Use this command to get online *help* text for DISCUTIL commands. If you specify a command name HELP displays text for that command. If you don't specify a command, you will see a list of all available commands and their syntax.

To invoke HELP enter:

```
discutil>HELP [ [CMD=] cmdname ]
```

For example, to see a list of all commands and their syntax, enter:

```
discutil>HELP
```

Or, to see information about the MOUNT command, enter:

```
:HELP MOUNT
```

LISTREDO

Use this command to display the command line history stack/queue, ordered from the least to the most recent command with absolute command reference numbers preceding each command. Use **LISTREDO** in conjunction with the **DO** command to reissue commands you previously entered.

To invoke **LISTREDO** enter:

```
discutil>LISTREDO
```

MOUNT

The **MOUNT** command is used to mount an unmounted volume. If the *ldev* represents a disk and if the device is responding, **DISCUTIL** will attempt to mount the volume. Only **MASTER** or **MEMBER** volumes can be mounted. Once mounted, a volume becomes available to **DISCUTIL** and is added to its mounted volume table. The *ldev* must be configured prior to mounting a volume on it.

To invoke **MOUNT** enter:

```
discutil>MOUNT [LDEV=] ldev
```

For example:

```
discutil>MOUNT 4
```

PDEV

The PDEV command displays the volume set name, volume number, and physical device path for all the disks known to DISCUTIL. All disks that are configured successfully, either at invocation or by the CONFIGURE command, are known to DISCUTIL. DISCUTIL reads the mounted volume table and uses the table information to configure the disks listed in the table.

The volume number is the volume's number in a volume set. For example, in a volume set of three volumes, each volume is given a number from one to three. This allows you to determine if you have all volumes of a volume set configured.

To invoke PDEV enter:

```
discutil>PDEV
```

The system responds with a display similar to:

CURRENT DEVICE CONFIGURATION:

```
LDEV : 1
  VOLUME SET NAME : MPEXL_SYSTEM_VOLUME_SET
  VOLUME NUMBER   : 1
  PATH            : 2/4.0.0
```

```
LDEV : 21
  VOLUME SET NAME : ACCOUNTING_VOL_SET
  VOLUME NUMBER   : 1
  PATH            : 6/4.0.0
```

```
LDEV : 22
  VOLUME SET NAME : ACCOUNTING_VOL_SET
  VOLUME NUMBER   : 2
  PATH            : 6/4.0.1
```

REDO

Use REDO to modify and re-execute a command in the command history stack.

To invoke REDO enter:

```
discutil>REDO [ [CMD=] cmid]
```

For example:

```
discutil>REDO 5
```

This command would re-execute the fifth command entered after allowing you to modify it.

To re-execute the last command entered enter:

```
discutil>REDO
```

SAVE

Use **SAVE** to retrieve files from disk and copy them to magnetic tape. You might use **SAVE** after a system failure, when a directory has been corrupted, or when a disk becomes inaccessible to the operating system.

If the volume label or the label table extent blocks are corrupt, **SAVE** will stop retrieving files from that disk and move onto the next.

Saving one set of files

You can tell **DISCUTIL** which files you want to copy to tape. The file(s) must reside on a disk that was mounted manually or during the startup of **DISCUTIL**.

To specify which files you want to save with **DISCUTIL**, do the following:

1. At the **discutil>** prompt, enter the **SAVE** command.
2. At the **ENTER FILE SET TO BE SAVED** prompt, specify the files you want copied using either MPE or HFS syntax. You can replace any part of the file set specification with **@** to indicate *all* members of the set.
3. At the prompt **TRAVERSE DIRECTORIES UNDER FILESET (Y/N)**, press **(RETURN)** to *not* save any hierarchical directories.
4. At the prompt **ENTER THE LDEV**, press **(return)** to tell the system to search all logical disk devices for the files. To search a single disk, enter its LDEV number.
5. At the prompt **ENTER THE VOLUME SET NAME**, press **(RETURN)** to search all disks. To search one set, specify the set.
6. At the prompt **ENTER THE MODIFICATION DATE (MM/DD/YYYY)**, enter a date to save only those files in the file set that have been modified on or since a certain date.
7. At the prompt **ENTER THE TAPE LDEV**, enter the LDEV number of the tape drive on which you want to store the files.

DISCUTIL searches the label tables of all *ldevs* specified for the indicated file(s). Each time a file is found, and when the file is successfully written to tape, it prints a message. You may also see other messages indicating that an error has occurred while **DISCUTIL** was saving a file.

If the end of a tape is reached before the procedure is complete, **DISCUTIL** will prompt you to mount a new tape. Once you have done so, **SAVE** continues until all specified files are copied. At that point, the **ENTER FILE SET TO BE SAVED:** prompt appears again so that you may specify additional files sets.

Saving multiple file sets

To save multiple sets of files, do the following:

1. Specify the first file set you want saved, answering each of the questions DISCUTIL asks. Refer to the preceding set of steps for information.
2. When you see the ENTER FILE SET TO BE SAVED prompt for the second time, press **RETURN** to complete the first tape set and rewind the tape.
3. At the `discutil>` prompt, enter the **SAVE** command again. (This begins the save procedure for the next file set.)

The following is a short sample of the interaction between a user and DISCUTIL during SAVE:

```
discutil>SAVE

*****
***** WARNING!!! *****
**** MANUALLY REWINDING THE TAPE DRIVE AND STARTING A NEW TAPESET ****
**** BEFORE THE CURRENT SAVE COMMAND IS COMPLETE WILL CAUSE FILES ****
**** TO BE LOST!!! TYPE (Return) at "ENTER FILE SET TO BE SAVED:"****
**** to complete the SAVE properly. SEE HELP "SAVE". *****
*****

ENTER FILE SET TO BE SAVED: MYFILE.JOHN.SMITH

TRAVERSE DIRECTORIES UNDER FILESET (Y/N)? n

ENTER THE LDEV: 3

ENTER THE MODIFICATION DATE(MM/DD/YYYY): 05/25/1994

ENTER THE TAPE LDEV: 7

MYFILE .JOHN .SMITH - LDEV 3 - ADDR $0002CA0 - FOUND
MYFILE .JOHN .SMITH - LDEV 3 - ADDR $0002CA0 - SAVED

*****
***** WARNING!!! *****
**** MANUALLY REWINDING THE TAPE DRIVE AND STARTING A NEW TAPESET ****
**** BEFORE THE CURRENT SAVE COMMAND IS COMPLETE WILL CAUSE FILES ****
**** TO BE LOST!!! TYPE (Return) at "ENTER FILE SET TO BE SAVED:"****
**** to complete the SAVE properly. SEE HELP "SAVE". *****
*****

ENTER FILE SET TO BE SAVED:
```

SHOWDEV

The **SHOWDEV** command displays information about each device that is currently in the system configuration. For each configured device, the device's *ldev*, *class*, and *physical* are displayed.

To invoke **SHOWDEV** enter:

```
discutil>SHOWDEV
```

The system responds with a display similar to the following:

1	DISC	2/4.0.0
3	DISC	2/4.0.2
7	TAPE	6/4.3.0
20	TERMINAL	2/4.1.0
21	DISC	6/4.0.0
25	DISC	2/8.0.0

TAPE

Use the **TAPE** command to display the *ldev* number of the *current* tape or to specify a new *current* tape. If the *ldev* parameter is omitted, the current tape *ldev* is displayed. If the *ldev* parameter is supplied, that *ldev* will become the new tape *ldev*. The *ldev* must have been configured as a tape.

To invoke **TAPE** enter:

```
discutil>TAPE [ [LDEV=] ldev]
```

For example, to see the *current* tape drive enter:

```
discutil>TAPE
```

To assign a logical device number as the *current* tape drive enter:

```
discutil>TAPE 7
```

UNCONFIG

Use the **UNCONFIG** command to remove a device from the system's configuration. This frees or releases the corresponding *ldev* and physical path, making them available for reassignment with the **CONFIGURE** command.

To invoke **UNCONFIG** enter:

```
discutil>UNCONFIG [LDEV=] ldev
```

For example:

```
discutil>UNCONFIG 14
```

DISCUTIL messages and error handling

DISCUTIL displays three types of messages:

- information messages, which inform the operator about the status of DISCUTIL
- warning messages, which generally occur when the operator has made an error during data input. Warning messages begin with **WARNING -** .
- error messages, which tell the Operator which errors DISCUTIL has encountered; typically, device errors. Error messages begin with **ERROR -** .

If a tape error occurs you must re-enter all file sets when prompted for a new tape unless the file sets were entered while a *previous* tape was mounted. If the tape error occurred while a *previous* tape was mounted DISCUTIL will automatically resave all files on the bad tape.

If an end-of-tape (EOT) is encountered in the middle of a file during DISCUTIL activity, the break in the file will be placed on a page boundary. If a tape write error occurs, DISCUTIL saves all files on the corrupt tape to a new tape. If part of that corrupt file is on the previous tape, the whole file is written to the new tape. Two end-of-file (EOF) marks are written to the tape any time DISCUTIL cannot save the entire file on a tape. VOLUTIL RECOVER is sensitive to this message and aborts that file's recovery at a double EOF before continuing.

Additional Discussion

For more information on the RECOVER command of VOLUTIL refer to *Volume Management Reference Manual* (32650-90045)

For more information on *resetting* the system refer to *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042).

EDIT/3000

The **EDIT/3000** text editor lets you create and edit ASCII files. Specifically, you may insert, delete, replace, modify and search for characters and strings of characters. You may run **EDIT/3000** in either of two modes, interactive or batch.

In an interactive session, you enter commands and text from an interactive terminal. Messages and other output (such as prompt characters) from **EDIT/3000** are listed on the terminal.

In batch mode, commands and text records are supplied through a batch input medium such as a jobstream or magnetic tape. Messages and output from **EDIT/3000** are listed on the standard output device, usually a line printer.

Operation

To invoke **EDIT/3000** enter in interactive mode, enter:

EDITOR

The system responds with a message similar to the one below and displays the **EDIT/3000** prompt, awaiting your commands:

```
HP32201A.07.17 EDIT/3000 Mon, Mar 28, 1994, 3:19 PM
(C) HEWLETT-PACKARD CO. 1985
/
```

For example, to create a new *work* file, you issue the **ADD** command and enter data. After the data is entered you use the **KEEP** command to save it as a permanent file. For example:

```
/ADD
 1
.
.
.
100 //
/KEEP filename,unn
```

To modify an existing file, use the **TEXT** command to open it in **EDIT/3000**, make the necessary modifications, and then use the **KEEP** command to save it. For example:

```

/TEXT FILE1
.
.
.
/KEEP FILE1
FILE1 ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
PURGE OLD? yes

```

Below is a summary of commands you may use with **EDIT/3000**.

Table 7-1. EDIT/3000 Commands

COMMAND	DESCRIPTION
ADD	Enters text into the WORK file from the standard input device and/or from the HOLD file.
BEGIN	Used as the first expression in a BEGIN-END pair.
CHANGE	Changes existing contents of the WORK file.
COPY	Copies text from one location to another in the WORK file.
DELETE	Deletes characters and/or lines from the WORK file.
END	Terminates EDIT/3000 operation. Or, when used with a matching BEGIN command, terminates a BEGIN-END pair.
FIND	Finds a specific position or a character string in the WORK file.
GATHER	Moves portions of text from one location to another in the WORK file and renumbers the lines. (The text is deleted from its original location.) Also can be used to renumber all lines in the WORK file.
HOLD	Copies part or all of the WORK file into the HOLD file for subsequent recopying into one or more locations of the WORK file.
INSERT	Inserts text into the WORK file from the INPUT file or from the HOLD file at a specific position.

Table 7-1. EDIT/3000 Commands (continued)

COMMAND	DESCRIPTION
JOIN	Copies all or part of the JOIN file to the WORK file.
KEEP	Saves all or part of the WORK file into an MPE/iX file.
LIST	Lists all or part of the WORK file to the OUTPUT file or to any other specified file.
MODIFY	Modifies text in the WORK file using one or more subcommands (DELETE, INSERT and REPLACE) of the MODIFY command.
NOT	Reverses a flag after executing the command immediately following the NOT command.
OR	Sets the flag true, or skips the OR command and the command immediately following it if the flag is already true.
PROCEDURE	Calls and executes a procedure previously written and stored in a segmented library (SL) file.
Q	Displays a user-defined message at the terminal.
REPLACE	Replaces one or more lines in the WORK file with new text from the standard input file or from the HOLD file.
SET	Alters EDIT/3000 default operating criteria.
TEXT	Copies the contents of a TEXT file into the WORK file, deleting the current WORK file contents.
USE	Instructs EDIT/3000 to receive commands from the USE file and to send messages to the OUTPUT file and, generally, to expect input from the INPUT file.
VERIFY	Reports the current EDIT/3000 operating conditions declared in a SET command, or the default conditions not declared in a SET command.
WHILE	Causes EDIT/3000 to repeat commands in a predefined command block.
XPLAIN	Lists an explanation of all or part of the EDIT/3000 commands.
YES	Sets a flag for a WHILE command block true.
Z::= or Z	Assigns the value of a character string variable to Z and uses that value whenever Z appears as a part or all of a command.
:	Instructs EDIT/3000 to pass the rest of the record to MPE/iX.

File Definitions

EDIT/3000 uses seven files: INPUT, OUTPUT, WORK, TEXT, JOIN, HOLD, and USE. Each file is described below.

INPUT	Used to enter commands and text records to EDIT/3000. Generally, this file is a terminal in interactive mode and a batch input device in batch mode. EDITIN is the formal file designator.
OUTPUT	Receives messages (and prompt characters in interactive sessions). Generally this file is a terminal in interactive mode and a line printer in batch mode. EDITOUT is the formal file designator.
WORK	Contains the information to be modified. When a file is created and text is added, or when an external file is copied into the EDIT/3000 subsystem for modification, the text is written into the WORK file and all modifications are performed on it. The WORK file may be saved under a new file name or to an existing file.
TEXT	An existing ASCII file copied into the WORK file with the TEXT command.
JOIN	All or a portion of an external file which is copied into the WORK file with the JOIN command. The information can be inserted into the WORK file at any point. The contents of the existing JOIN file are not altered by the JOIN command.
HOLD	A temporary file that is generally used for holding interim information.
USE	An external user file containing EDIT/3000 commands and, optionally, text records which is called with a USE command. When a USE command is issued, all commands are read from the USE file and any EDIT/3000 messages are sent to the OUTPUT file.

Additional Discussion

For more information refer to *EDIT/3000 Reference Manual* (03000-90012).

FCOPY

You use the **FCOPY** utility to copy and translate files. The **FCOPY** command identifies a *from* file, a *to* file, and one or more **FCOPY** functions. Some of the functions you may perform include:

- converting data
- copying files from other systems
- appending files
- extracting subsets of files
- displaying binary files in **ASCII** format
- copying byte-stream files

A *from* file is the input file for an **FCOPY** command; it contains the data you want to copy. A *to* file is the output file to which you want to copy the data. They are the only two options that you must specify.

To accommodate the introduction of POSIX in MPE/iX, the **FCOPY** utility lets you copy files from HFS directories into accounts and groups and into other HFS directories. Users with SM capabilities can use **FCOPY** to copy files to MPE accounts outside of their logon account.

Keep in mind the following points when using **FCOPY** to work with HFS files:

- If the *from* file or *to* file name begins with an asterisk (*), the file equation can resolve a filename in HFS syntax that begins with a dot(.) or a slash (/).
- If you are copying files to an HFS directory, you must first use the **FILE** command and specify the file name in HFS syntax
- You cannot use **FCOPY** to copy directories. If the *from* file or *to* file is a directory, you will see an error message.
- You cannot copy compatibility mode (CM) files to HFS directories.

Operation

To invoke FCOPY enter:

FCOPY FROM=*input file*;TO=*output file*;*functions*

For example, to create a new disk file (in exactly the same format as another file) use the NEW parameter:

> FROM=OLDFILE;TO=NEWFILE;NEW

To copy a *subset* of one file to another enter:

> FCOPY FROM=FILEONE;TO=FILETWO;SUBSET=29:33

In this example FCOPY copies the *30th through 34th* records. You specify ;SUBSET=29:33 because FCOPY sees the first record in a file as *record number 0*.

To copy a tape created in EBCDIC format in an IBM environment with a label of VOL 000001,IBM you would enter the following:

FILE T;DEV=TAPE;LABEL=000001,IBM;REC=-132,20,F,ASCII
FILE HPFILE;REC=-132,1,F,ASCII;DISC=12000
FCOPY FROM=*T;TO=*HPFILE;NEW;EBCDICIN

Be sure the parameters in the file equations (blocking factor, record size, etc.) are correct.

To copy a file to your terminal screen in *hexadecimal* format enter:

FCOPY FROM=FILEONE;TO=;HEX

If you specify ;TO= without anything after it the default is the \$STDLIST device.

“From” and “To” Files

A *from* file is the input file for an FCOPY command; it contains the data you want to copy. A *to* file is the output file to which you want to copy the data.

Identify a *from* file for an FCOPY command with the FROM parameter. FROM has the following format:

[={*fromfile*}]
FROM [={ * }]
[={<*empty*>}]

The value you assign to FROM can be either an input file name (*fromfile*), an asterisk (*), or nothing at all (<*empty*>). An asterisk backreferences the fromfile named in a file equation. Leaving FROM empty lets you use your terminal (or a spoolfile during a job) as the input file.

In the example below, the **FROM** parameter describes an input file named *input* for an **FCOPY** command.

FROM=*input*

Specify a *to* file with the **T0** parameter. **T0** has the following format:

```
[={tofile      }]  
;T0 [={*      }]  
    [={<empty >}]
```

The value that you assign to **T0** can be either the name of the output file (*tofile*), an asterisk (*), or nothing at all (<*empty* >). An asterisk backreferences a file named in a file equation. Leaving **T0** empty lets you copy files to the **\$STDLIST** device.

The example below describes an output file name *outfile* for an **FCOPY** command.

;T0=*outfile*

FCOPY Functions

In addition to specifying input and output files in an **FCOPY** command, you also describe the **FCOPY** functions that you want to perform. Each function has its own syntax and guidelines for its use. The **NEW** function, for example, lets you create a new disk file. An example of an **FCOPY** command using the **NEW** function is:

FCOPY
>FROM=*oldfile***;T0=***newfile***;NEW**

The table below lists **FCOPY** functions and their descriptions in alphabetical order.

Table 8-1. FCOPY Functions	
Function	Description
BCDICIN	Translates from BCDIC to ASCII.
BCDICOUT	Translates from ASCII to BCDIC.
CCTL	Designates the first character of each record as a carriage control character in the “to” file.
CHAR	Displays the contents of a file as ASCII characters.
CLEAR	Displays the contents of a file as character codes.
COMPARE	Compares two files.
DEBLOCK	Deblocks blocked records.
EBCDICIN	Translates from EBCDIC to ASCII.
EBCDICOUT	Translates from ASCII to EBCDIC.
EBCDIKIN	Translates from ECDIC to JIS.
EBCDIKOUT	Translates from JIS to EBCDIK.
FILES	Copies multiple file from tape.
HEX	Displays the contents of a file in hexadecimal form.
HEXO	Displays the contents of a file in hexadecimal form, and the sequential record number in octal form.
IGNERR	Bypasses and reports magnetic tape errors.
KANA	Displays the contents of a file as JIS character symbols.
KEY	Specifies a key sequence in which to copy a KSAM file.
NEW	Creates a new permanent disk file.
NOCCTL	Specifies that the first character of each record in the “from” file will not be a carriage control character.
NOKSAM	Copies a <i>compatibility mode</i> KSAM data file to a non-KSAM file.
NOUSERLABELS	Omits user labels when copying between disk and tape.
OCTAL	Displays the contents of a file in octal form.
SKIPEOF	Positions a serial storage device at a desired file.
SUBSET	Copies a subset of a file.
UPSHIFT	Converts lowercase characters to uppercase.
VERIFY	Compares files after copying.

Defining Files

If you are copying files from or to devices other than disk, you must define the files and their associated devices with the **FILE** command before issuing an **FCOPY** command. For example, to copy a file from magnetic tape to a line printer, define two device files as follows:

```
FILE TAPEFILE;DEV=TAPE;REC=-80,25,F,ASCII  
FILE PRINTER;DEV=LP
```

TAPEFILE and **PRINTER** are the formal file designators you use in the **FCOPY** command. **TAPE** and **LP** are device class names for a magnetic tape unit and a line printer respectively. Device class names are defined when the system is configured and may vary from one installation to another.

You may use the two formal file designators as the “from” and “to” files in an **FCOPY** command. Type an asterisk (*) before each file name to tell **FCOPY** to refer to the previous **FILE** command for the file’s description. The **FROM** and **TO** parameters below reference the two files defined above:

```
FROM=*TAPEFILE;TO=*PRINTER
```

FCOPY assumes files to have default characteristics unless you define the files with other characteristics. For more information on the **FILE** command, refer to the *MPE/iX Commands Reference Manual* (32650-90003).

General Guidelines for FCOPY Commands

An **FCOPY** command must follow these general guidelines:

- Semicolons always separate the different components of a command. You can leave spaces between components. However, there can be no more than 70 characters between two semicolons. For example, all three **FCOPY** commands below are valid.

```
FROM=A;    TO=B; NEW  
FROM=C;TO=D;NEW  
FROM=E; TO=F; NEW
```

- To continue an **FCOPY** command onto more than one line, use an ampersand (&) at the end of each line except the last. An **FCOPY** command has no maximum length. However, a single line of an **FCOPY** command cannot be more than 72 characters long. For example:

```
FROM=A; &  
TO=B; &  
NEW
```

Using FCOPY with KSAM Files

To FCOPY from an *old compatibility mode* KSAM file to a *new compatibility mode* KSAM file enter:

```
FCOPY FROM=OLDFILE;TO=(DATAFILE,KEYFILE)
```

OLDFILE is the *old* compatibility mode KSAM data file. DATAFILE is the *new* compatibility mode data file and KEYFILE is the *new* compatibility mode key file. The new key and data files are constructed for you with exactly the same structure as the old key and data files. The ;NEW option need *not* be used.

To FCOPY from an MPE/iX file (a *flat* file) to an old KSAM file (compatibility or native mode) enter:

```
FCOPY FROM=FLATFILE;TO=KSAMFILE
```

To FCOPY from a KSAM file to an old flat file enter:

```
FCOPY FROM=KSAMFILE;TO=ANYFILE
```

KSAMFILE is either a native mode KSAM file or compatibility mode KSAM data file. ANYFILE is an old MPE/iX file.

To FCOPY from a *compatibility mode* KSAM file, but to treat the file as if it were an MPE/iX flat file, enter:

```
FCOPY FROM=KSAMFILE;TO=ANYFILE;NOKSAM
```

You may use the NOKSAM parameter with only *compatibility mode* KSAM files.

To FCOPY from *any* old KSAM file (compatibility or native mode) to a *new* native mode KSAM file enter:

```
FCOPY FROM=OLDFILE;TO=(NEWFILE)
```

OLDFILE is the old KSAM file. NEWFILE is the *new* native mode KSAM file.

Additional Discussion

For more information refer to *FCOPY Reference Manual* (32212-90003).

FSCHECK

The file system check utility (**FSCHECK**) is a native mode program used to detect and repair inconsistencies found in the file directories and file label tables of the MPE/iX operating system and to display the file extent map. It is a *standalone* utility and should be the only program running on the system when it is in use.

FSCHECK also provides a consistent MPE/iX subsystem interface, including **LISTREDO**, **REDO**, **DO**, **USE**, **DEBUG**, and **LOG** functionality.

Warning

**Do not use this utility without proper service center support.
Unauthorized use will void your warranty and may cause data loss!**

Operation

There are two parts to the **FSCHECK** utility, the program and the message catalog. The program can reside in any group and account, but the message catalog must reside in **MPEXL.TELESUP**. Each version of the program has a unique message catalog, and the catalog contains a version ID to prevent the mixing of incompatible programs and catalog versions. If the message catalog is not in **MPEXL.TELESUP**, use a file equation to redirect it to the correct file.

To execute the **FSCHECK** utility, type **FSCHECK** at the MPE/iX prompt. You can also invoke **FSCHECK** with the MPE/iX **RUN** command using the **INFO** parameter to pass commands. The group and account in **FSCHECK** that is being run should have **PM**, **MR**, **DS**, and **PH** capabilities.

The **FSCHECK** utility uses the formal file designators **FSCHKIN** and **FSCHKOUT** for input and output respectively. The default input file is **\$STDINX** and the default output file is **\$STDLIST**, though you can use file equations to redirect them to other files.

On the following pages, each of the **FSCHECK** commands is listed in alphabetical order. String sequences (tokens) in brackets next to the command name indicate abbreviations for the command.

CHECKDIRC [CD]

The CHECKDIRC command checks the directory on the specified volume set for internal consistency and makes sure that for each directory entry there exists an associated file label entry. This command assumes that all volumes of the identified volume set are mounted and available and that the system is fully operational.

Syntax

```
CHECKDIRC    [DEV=] set_name
              ALL
              [;IGNORE]
              [;FIX]
              [;LOG=filename]
```

Parameters

<i>set_name</i>	<i>Set_name</i> is the name of the volume set whose directory is to be checked. The volume set must be opened (the set's master volume must be mounted in the MASTER state) as displayed by the DSTAT command.
ALL	All checks the directory on all mounted volume sets.
IGNORE	Ignores errors detected by CHECKDIRC and continues checking the directories.
FIX	Fixes the errors detected by CHECKDIRC and continues checking the directories. This is the default option.
<i>filename</i>	<i>Filename</i> is the name of a log file on which CHECKDIRC messages are to be written. If this parameter is omitted, the output will be displayed only to \$STDLIST.

Example

```
fscheck:CHECKDIRC ALL Return
```

CHECKEXTENTS [CE]

The CHECKEXTENTS command checks the extent map for each file label in the label table for duplicate extent descriptors. Duplicate extent descriptors are those with overlapping file sector offset.

Syntax

```
                                ldev  
CHECKEXTENTS  [DEV=] set_name  
                                ALL
```

Parameters

The *DEV* parameter is optional. If omitted, the default is to check the extent maps on each mounted volume of the system volume set.

ldev *Ldev* is a number from 1 to 32767, specifying the logical device on which the volume is mounted. The volume must be mounted in the MASTER or MEMBER state as displayed by the DSTAT command

set_name *Set_name* is the name of a mounted volume set whose master volume is mounted in the MASTER state as determined by the DSTAT command. The extent map check is performed on each mounted volume of the set.

ALL ALL checks the extent maps on each volume mounted in the MASTER or MEMBER state.

Example

```
fscheck: CHECKEXTENTS ALL return
```

CHECKLABEL [CL]

The CHECKLABEL command checks the label table(s) on the specified volume(s) for internal consistency and verifies that each file label entry has an associated entry in the volume set's directory. It does not verify the integrity of HFS-syntax files whose file names are blanked out in the file label.

If neither the CNAME or VNAME parameter is specified, the label table of each volume in the sytem volume set is checked by default (MPEXL_SYSTEM_VOLUME_SET).

Syntax

```
CHECKLABEL  [DEV=] [ldev]  
                                [set_name]  
                                [ALL]  
                                [;IGNORE]  
                                [;FIX]  
                                [;ASK]  
                                [;LOG=filename]
```

Parameters

<i>ldev</i>	<i>Ldev</i> is a number from 1 to 32,767, specifying the logical device on which the volume to be checked is mounted. The volume must be mounted in the master or member state, as determined by the DSTAT command.
<i>set_name</i>	<i>Set_name</i> is the name of the volume set whose label tables are to be checked. The volume set must be opened (the set that is master must be mounted in the MASTER state, as displayed by the DSTAT command).
ALL	ALL checks the label tables on each volume mounted in the MASTER or MEMBER state as determined by the DSTAT command.
IGNORE	Ignore errors detected by CHECKLABEL and continue checking the labels.
FIX	Fixes the errors detected by CHECKLABEL and continues checking the labels.
ASK	ASK prompts the user to choose to fix the errors or not. This is the default option.
<i>filename</i>	<i>Filename</i> is the name of a log file on which CHECKLABEL messages are to be written. If this parameter is omitted, the output will be displayed to \$STDLIST .

Example

```
fscheck: CHECKLABEL ALL Return
```

CHECKFILE [CF]

The **CHECKFILE** command checks the label of the specified file for internal consistency.

Syntax

```
CHECKFILE [FILENAME=] filename
```

Parameters

<i>filename</i>	Specifies the name of the file to be checked. The file name can be an MPE/iX file name of the form <i>filename</i> [.group][.account] or it can be a fully-qualified HFS pathname. You may use wildcard characters that conform to MPE/iX convention, to specify the file, group, and account names.
-----------------	--

Examples

```
fscheck: CHECKFILE @.PUB.SYS return
```

```
fscheck: CHECKFILE /mydir/myfile1 return
```

CHECKALL [CA]

The CHECKALL command performs the CHECKLABEL, CHECKDIRC, and CHECKEXTENTS commands consecutively.

Syntax

```
CHECKALL  [DEV=] set_name
          ALL
```

Parameters

The DEV parameter is optional. If omitted, the default is to check the directory, labels, and extent maps on each mounted volume of the system volume set.

set_name *Set_name* is the name of a mounted volume set whose master volume is mounted in the MASTER state as determined by the DSTAT command. The label, directory, and extent map check is performed on each mounted volume of the set.

ALL ALL checks the labels, directory, and extent maps on each volume mounted in the MASTER or MEMBER state.

Example

```
fscheck: CHECKALL ALL return
```

DEBUG

The DEBUG command invokes the native mode system.

Syntax

```
DEBUG
```

Parameters

None

Example

```
fscheck: DEBUG return
```

DISPLAYLABEL [DL]

The DISPLAYLABEL command displays the fully qualified file name and offset within the label table for each file label entry of the label table on the specified logical device.

Syntax

```
DISPLAYLABEL  [DEV=] ldev
               set_name
               ALL
```

Parameters

DEV=	DEV= is an optional keyword for the parameter. If it is omitted, the default is to display the contents of the label table for each mounted volume of the system volume set.
<i>ldev</i>	<i>Ldev</i> is a number from 1 to 32,767, specifying the logical device on which the volume is mounted. The volume must be mounted in the master or member state, as displayed by the DSTAT command.
<i>set_name</i>	<i>Set_name</i> is the name of a mounted volume set whose master volume is mounted in the MASTER state as displayed by the DSTAT command. For each mounted volume in the set, the contents of the label table is displayed.
ALL	ALL displays the contents of the label table for each volume mounted in the MASTER or MEMBER state, as determined by the DSTAT command.

Example

```
fscheck: DISPLAYLABEL DEV=1 Return
```

DISPLAYEXTENTS [DE]

This command displays the extent map for the specified file. For each extent of the file, it displays the number of sectors in the extent, the single vector disk sector address (in hex), the file sector offset (in hex), and the volume set index of the volume on which the extent resides.

Currently the file name must be fully qualified, that is, with the group and the account names specified.

Syntax

```
DISPLAYEXTENTS [FILENAME=] filename
```

Parameters

<i>filename</i>	Specifies the name of the file whose extent map you want displayed. The file name can be an MPE/iX file name of the form <i>filename</i> [. <i>group</i>][. <i>account</i>] or it can be a fully-qualified HFS pathname.
-----------------	--

Examples

```
fscheck: DISPLAYEXTENTS NL.PUB.SYS Return
```

```
fscheck: DISPLAYEXTENTS /sys/mydir1/myfile Return
```

DISPLAYLOCKFILE [DLF]

Displays the fully qualified file name of all of the locked files on the specified volume. Files whose extents contain bad sectors are being locked.

Syntax

```
                                ldev  
DISPLAYLOCKFILE [DEV=] set_name  
                                ALL
```

Parameters

The DEV parameter is optional. If omitted, the default is to display the locked files on each mounted volume of the system volume set.

<i>ldev</i>	<i>Ldev</i> is a number from 1 to 32767, specifying the logical device on the locked files on the volume to be displayed, is mounted. The volume must be mounted in the MASTER or MEMBER state as determined by the DSTAT command.
<i>set_name</i>	<i>Set_name</i> is the name of the volume set whose locked files are to be displayed. The volume set must be opened (the set's master must be mounted in the MASTER state, as determined by the DSTAT command).
ALL	ALL displays the locked files on each volume mounted in the MASTER or MEMBER state as determined by the DSTAT command.

Example

```
fscheck: DISPLAYLOCKFILE ALL (return)
```

DO

This command re-executes a command from the command line history stack. It is identical in function to the command DO.

Syntax

```
DO [CMD=] cmd_id
```

Parameters

<i>cmd_id</i>	<i>Cmd_id</i> identifies a particular command in the command line history stack. It can be a number relative to the last command (<i>-n</i>), an absolute number (<i>n</i>) identifying the particular command, or a string used to match a particular command in the stack.
---------------	--

Example

```
fscheck: DO CMD=-2 (Return)
```

EMPTYSLOUGH [ES]

Removes all extents from the slough file without attempting to deallocate the secondary storage owned by its extents. This command is used for file corruption work-around when inconsistencies exist between the free space map and the label table such that there exist extent descriptors for which the corresponding secondary storage is not permanently allocated.

Syntax

```
EMPTYSLOUGH  ldev
               [DEV=] set_name
               ALL
```

Parameters

The DEV parameter is optional. If omitted, the default is to empty the slough file on the system volume set.

<i>ldev</i>	<i>ldev</i> is a number from 1 to 32,767, specifying the logical device on which the volume is mounted. The volume must be mounted in the MASTER state as displayed by the DSTAT command.
<i>set_name</i>	<i>Set_name</i> is the name of a mounted volume set whose master volume is mounted in the MASTER state as determined by the DSTAT command.
ALL	ALL empties the slough file on each volume set as mounted in the MASTER state.

Example

```
fscheck:  EMPTYSLOUGH 1 return
```

EXIT [E]

This command terminates the FSCHECK utility and returns to the process from which it was invoked.

Syntax

```
EXIT
```

Parameters

None

Example

```
fscheck:  EXIT Return
```

EXTENTDISTRIB [ED]

Displays the distribution of extents by extent size or file size on the specified volume(s). If you choose the **EXTENT** option, **FSCHECK** displays the total number of extents that falls in the extent size range. If you choose the **FILE** option, **FSCHECK** displays the total number of files, and the average number of extents per file that falls in the file size range.

Syntax

```
                                ldev
EXTENTDISTRIB [DEV=]set_name
                                ALL
                                [;EXTENT]
                                [;FILE  ]
```

Parameters

The DEV parameter is optional. If omitted, the default is to display the extent distribution on each mounted volume of the system volume set.

<i>ldev</i>	<i>ldev</i> is a number from 1 to 32,767, specifying the logical device on which the extent distribution on the volume to be displayed is mounted. The volume must be mounted in the MASTER or MEMBER state as determined by the DSTAT command.
<i>set_name</i>	<i>Set_name</i> is the name of the volume set whose extent distribution are to be displayed. The volume set must be opened (the set's master must be mounted in the MASTER state as displayed by the DSTAT command.
ALL	ALL displays the extent distribution on each volume mounted in the MASTER or MEMBER state as determined by the DSTAT command.
EXTENT	EXTENT displays the distribution of extent by extent size. This is the default option.
FILE	FILE displays the distribution of extent by file size.

Example

```
fscheck:  EXTENTDISTRIB ALL return
```

HELP

This command provides a list of the **FSCHECK** utility commands and a description of the function of each.

Syntax

HELP

Parameters

None

Example

HELP [Return](#)

LISTREDO

This command displays the contents of the command line history stack, from the least recently entered command to the most recently entered command. It is identical in function to the **LISTREDO** command.

Syntax

LISTREDO

Parameters

None

Example

fscheck: LISTREDO [Return](#)

LOG

This command logs the user or program dialog to the specified file, as it appears to the user.

Syntax

LOG [FILENAME=] *filename*

Parameters

filename *Filename* is any valid MPE/iX file name to which the user has read/write access. It cannot be an HFS-syntax file. If the file doesn't exist, it will be created.

Example

fscheck: LOG FILENAME=FSCHKLOG [Return](#)

PURGEFILE [PF]

This command purges the specified file.

Syntax

```
PURGEFILE [FILENAME=]filename
```

Parameters

filename Specifies the file to be purged. The file name may be an MPE/iX file of the form *filename*[*.group*][*.account*], or it may be a fully qualified HFS pathname.

Examples

```
fscheck: PURGEFILE TEMP.PUB.SYS Return
```

```
fscheck: PURGEFILE /sys/dir1/myfile Return
```

REDO

This command edits and re-executes a command from the command line history stack. It is identical in function to the **REDO** command.

Syntax

```
REDO [CMD=] cmd_id
```

Parameters

cmd_id *Cmd_id* identifies a particular command in the command line history stack. It can be a number relative to the last command (*-n*), an absolute number (*n*) identifying the particular command, or a string used to match a particular command in the stack.

Example

```
fscheck: REDO CMD=5 Return
```

SYNCAccounting [SA]

This command synchronizes the account and group disk space accounting with the disk space information found in the file labels of all files on a specified volume set. For system volume sets containing HFS directories, disk space accounting is done for the account and group structure only.

After performing **SYNCAccounting**, the information reported by the **REPORT** command will coincide with the information reported by the **LISTF** command.

Syntax

```
SYNCAccounting [DEV=] [set_name] [;QUIET]
```

Parameters

The DEV parameter is optional. If omitted, the default is to synchronize the account and group directories of the system volume set.

set_name *Set_name* is the name of the volume set whose directories are to be synchronized. The volume set must be opened. (The set's master volume must be mounted in the MASTER state as displayed by the **DSTAT** command).

QUIET **QUIET** specifies that the list of accounts and groups processed should not be displayed on the **\$STDLIST** device. If **;QUIET** is not specified **SYNCAccounting** displays a list of accounts and groups as they are processed. The format of this list is similar to the output of the **REPORT** command. When the disk space accounting is corrected for a group or account **ACCOUNTING CORRECTED** is displayed next to the group or account entry in the list.

Example

```
fscheck: SYNCAccounting MPEXL_SYSTEM_VOLUME_SET return
```

TOTALExtENTS [TE]

This command displays the total number of sectors occupied by directory space, special space, permanent file space, spool file space, and new and temp file space on the specified volume(s).

Syntax

```
                                ldev  
TOTALExtENTS [DEV=] set_name  
                                ALL
```

Parameters

The DEV parameter is optional. If omitted, the default is to display the extent totals on each mounted volume of the system volume set.

<i>ldev</i>	<i>Ldev</i> is a number from 1 to 32,767, specifying the logical device on which the extent totals on the volume to be displayed is mounted. The volume must be mounted in the MASTER or MEMBER state as determined by the DSTAT command.
<i>set_name</i>	<i>Set_name</i> is the name of the volume set whose extent totals are to be displayed. The volume set must be opened (the set's master must be mounted in the MASTER state as displayed by the DSTAT command).
ALL	ALL displays the extent totals on each volume mounted in the MASTER or MEMBER state, as determined by the DSTAT command.

Example

```
fscheck: TOTALEXTENTS ALL return
```

UNLOCKFILE [UF]

MPE/iX locks files whose extents contain bad sectors. This command unlocks a locked file so that it can be purged or analyzed.

Syntax

```
UNLOCKFILE [FILENAME=] filename
```

Parameters

<i>filename</i>	<i>Filename</i> is the fully qualified name of the file to be unlocked.
-----------------	---

Example

```
fscheck: UNLOCKFILE AFILE.BGROUP.CACCT return
```

USE

This command reads and executes commands contained in the specified file.

Syntax

```
USE [FILENAME=] filename
```

Parameters

<i>filename</i>	<i>Filename</i> is any valid MPE/iX file name. The file must already exist, and the user must have read access.
-----------------	--

Example

```
fscheck: USE FILENAME=FSCHKUSE Return
```


I7DB8CNV

I7DB8CNV converts the character data in an IMAGE database from any Hewlett-Packard 7-bit national substitution set to ROMAN8. The program is a special version of the DBLOAD.PUB.SYS program, and the conversion is done as part of a database load.

Operation

To run I7DB8CNV, do the following:

1. Enter RUN DBUNLOAD.PUB.SYS to unload the database to tape:
2. Enter RUN DBUTIL.PUB.SYS,ERASE to erase the database data.
3. Enter RUN I7DB8CNV to convert the data and reload it into the database.

I7DB8CNV will request the following information:

- The 7-bit national substitution set where the conversion is to be made.
- The database name.
- Whether or not to convert all data fields of type X or U. Enter YES or to convert the data fields. Otherwise, enter NO and follow the prompts to specify each field of type U or X.

The single field in an automatic data set is not proposed for conversion. Whether or not its values are converted depends on the response to the item(s) it is linked to for detail data set(s). At the end of each data set, the user is asked to confirm that the correct fields to be converted from that data set have been selected. Again, a is a YES answer, an N allows the user to change the data fields in the data set to be converted.

I7DB8CNV then loads the database from tape. As each record is read, those fields which were selected have their data converted according to the algorithm for the 7-bit national substitution set selected at the beginning of the program.

I7DB8CNV will not allow 8-bit data (bytes with the high-order bit set) in the data fields it is trying to convert. The utility will not abort, but the field in question will not be converted, and the following warning will be issued:

8-bit data encountered in item [*itemname* in DS data set]

If the program should abort for any reason during the conversion, the user must log on again to clear the temporary files used during the conversion process before running the program again.

Following is the dialog from a sample run of the I7DB8CNV utility.

RUN I7DB8CNV.PUB.SYS

HP European 7-bit character sets are:

1. SVENSK/SUOMI
2. DANSK/NORSK
3. FRANCAIS
4. FRANCAIS M
5. DEUTSCH
6. UK
7. ESPANOL
8. ESPANOL M
9. ITALIANO

From which character set should conversion be done: 2

WHICH DATA BASE: QWERTZ

Convert all fields of type U,X in all data sets (Y/N)? N

Data Set SET1 fields to be converted:

ITEM1 (Y/N)?

ITEM2 (Y/N)?

ITEM3 (Y/N)? N

ITEM4 (Y/N)?

Is Data Set SET1 correctly defined (Y/N)?

Data Set SET2 - Automatic Master

Data Set SET3 fields to be converted:

ITEM1 (Y/N)?

ITEM5 (Y/N)? N

ITEM6 (Y/N)? N

Is Data Set SET3 correctly defined (Y/N)?

DATA SET 1: 19 ENTRIES

DATA SET 2: 0 ENTRIES

DATA SET 3: 25 ENTRIES

END OF VOLUME 1, 0 READ ERRORS RECOVERED

DATA BASE LOADED

END OF PROGRAM

:

LANGINST

The **LANGINST** utility builds and modifies the **LANGDEF.PUB.SYS** file, which contains all language-dependent information for every language that the system manager plans to configure. **LANGINST** gathers data from the **NLSDEF.PUB.SYS** and **CHRDEFxx.PUB.SYS** files and writes it to **LANGDEF.PUB.SYS**.

The system manager uses the **LANGINST** utility to:

- Add a language to the configuration file.
- Remove a language from the configuration file.
- Display and modify local formats of a configured language.
- Display the languages supported by Hewlett-Packard.
- Display the languages currently configured.
- Modify the system default language.

Any changes you make to **LANGDEF.PUB.SYS** will only be implemented if you perform a **COOLSTART** or **WARMSTART**.

Operation

The system manager initiates the **LANGINST** utility by entering:

LANGINST

When the **LANGINST** main menu appears, you choose one of the following functions:

0. EXIT
1. ADD LANGUAGE TO LANGDEF
2. DELETE LANGUAGE FROM LANGDEF
3. MODIFY NATIVE FORMATS
4. LIST HP SUPPORTED LANGUAGES
5. MODIFY THE SYSTEM DEFAULT LANGUAGE
6. LIST LANGUAGES CURRENTLY CONFIGURED
7. DISPLAY TRANSLATION TABLES

Listing Supported Languages

To list languages that are configurable on the system, select option 4 from the **LANGINST** main menu. You'll see a display like the one below.

HP SUPPORTED LANGUAGES:

0	NATIVE-3000	using	USASCII
1	AMERICAN	using	ROMAN8
2	CANADIAN-FRENCH	using	ROMAN8
3	DANISH	using	ROMAN8
4	DUTCH	using	ROMAN8
5	ENGLISH	using	ROMAN8
6	FINNISH	using	ROMAN8
7	FRENCH	using	ROMAN8
8	GERMAN	using	ROMAN8
9	ITALIAN	using	ROMAN8
10	NORWEGIAN	using	ROMAN8
11	.	.	.
12	.	.	.
13	.	.	.

press any key to continue ...

Adding a Language

To add a language, select option 1 from the **LANGINST** main menu. When the **Add Language to LANGDEF** menu is displayed, enter:

1. The language name or language ID number.
 - The addition is aborted if one of the following conditions is met:
 - ☐ Entering a language that is already configured on the system.
 - ☐ Entering a language not supported by NLS.
 - ☐ Pressing **Return**.
 - The addition continues if a language is requested that is supported but has not been previously configured. **LANGINST** configures the language and displays a message.
2. When the addition is completed, the **LANGINST** main menu is displayed.

Deleting a Language

LANGINST allows the system manager to delete any configured language except NATIVE-3000 (which is hard-coded and therefore always configured) and the system default language.

To delete a language, select option 2 from the LANGINST main menu. When the Delete Language from LANGDEF menu is displayed, enter:

1. The language name or language ID number.
 - The deletion is aborted if one of the following conditions is met:
 - Entering a language that is not configured on the system.
 - Entering the system default language.
 - Pressing **Return**.
 - The deletion continues if the language requested is configured but is not the system default language.
2. When the deletion is completed, the main menu is displayed.

Modifying Local Formats

As the system manager, you can modify the following local formats for any configured language:

- Date format (dateline format).
- Custom date format (short).
- Time format.
- Currency sign/name.
- Decimal and thousands indicator.
- Month names.
- Abbreviated month names.
- Weekday names.
- Abbreviated weekday names.
- Yes/no indicators.
- Direction of text.
- ASCII/EBCDIC translation tables.
- National date table.

If the language supports a special national table containing date information (such as KATAKANA), the last option lets the user modify the date information.

To modify local language formats, select option 3 from the LANGINST main menu. When the Modify Native Formats menu is displayed, enter:

1. The language name or language ID number.
 - The modification is aborted if one of the following conditions is met:
 - ☐ Entering a language that is not configured on the system.
 - ☐ Entering NATIVE-3000.
 - ☐ Pressing **Return**.
 - The modification continues if the language requested is configured.
2. If a configured language is entered, the user dialog is displayed:
 1. Long calendar format
 2. Date format (Calendar format)
 3. Custom date format (Short)
 4. Time format (Clock format)
 5. Currency sign
 6. Currency name
 7. Decimal and thousands separator
 8. Alternate numeric format
 9. YES and NO equivalents
 10. Month names.
 11. Month name abbreviations
 12. Weekday names
 13. Weekday name abbreviations
 14. Direction of text
 15. ASCII/EBCDIC translation tables
 16. Handle truncation in date format
 17. Process the national date table
 18. 16-bit Asian ASCII to EBCDIC translation table
 19. 16-bit Asian EBCDIC to ASCII translation table

```
Enter selection number      : 5
Business Currency sign     : F
Enter the new value        : Return
Fully qualified Currency sign : FF
Enter the new value        : Return
The currency sign currently follows the number, 100DM.
```

The following currency codes are available:

<CR> to retain the existing value.

0 - The currency symbol precedes the number, \$100.00.

1 - The currency symbol succeeds the number, 100.00DM.

2 - The currency symbol replaces the decimal point, 100\$00.

Enter the required currency codes (0, 1, or 2) :

There are to be no blanks before or after the currency symbol.

The following blank-control codes are available:

<CR> to retain the existing value.

0 - No blanks before or after the currency symbol.

1 - A blank is to precede the currency symbol.

2 - A blank is to succeed the currency symbol.

3 - A blank is to precede and succeed the currency symbol.

Enter the required code (0, 1, 2, or 3):

After you make a selection, the current value is displayed and you are prompted for a new value. If you enter a new value, LANGINST validates and replaces the old value with the current one.

After you have made changes to the file, a new copy of it is saved under the name LANGDEF and the old (unchanged) version of the file is saved under the name LANGD*xxx*. The number *xxx* increases by one each time a new copy of LANGDEF is saved. This allows the user to return to a configuration that existed before LANGDEF was changed. To return to the previous configuration, PURGE or RENAME the current LANGDEF then RENAME the LANGD*xxx* with the highest number LANGDEF.

Modifying ASCII/EBCDIC Translation Tables

To modify the ASCII/EBCDIC translation tables for any language other than NATIVE-3000, select option 3 from the LANGINST main menu. When the Modify Native Formats menu is displayed, enter:

1. The language name or language ID number.
 - The modification is aborted if one of the following conditions is met:
 - ☐ Entering a language that is not configured on the system.
 - ☐ Entering NATIVE-3000.
 - ☐ Pressing **Return**.
 - The modification continues if the language requested is configured.
2. If a configured language is entered, the following user dialog is displayed:
 1. Long calendar format
 2. Date format (Calendar format)
 3. Custom date format (Short)
 4. Time format (Clock format)
 5. Currency sign
 6. Currency name
 7. Decimal and thousands separator
 8. Alternate numeric format
 9. YES and NO equivalents
 10. Month names.
 11. Month name abbreviations
 12. Weekday names
 13. Weekday name abbreviations
 14. Direction of text
 15. ASCII/EBCDIC translation tables
 16. Handle truncation in date format
 17. Process the national date table
 18. 16-bit Asian ASCII to EBCDIC translation table
 19. 16-bit Asian EBCDIC to ASCII translation table

Enter selection number : 15

Input ROMAN8 character to be changed (HEX please) : 04
The current EBCDIC value is : 37
Enter the new EBCDIC value : 44
The ROMAN8 to EBCDIC table was updated
The EBCDIC to ROMAN8 table will be updated too
ASCII/EBCDIC table inconsistent for 44 <== 04,C8 (*)
The tables are inconsistent for ROMAN8 character C8 (**)
The current EBCDIC value is : 44
Enter the new EBCDIC value : 37
The ROMAN8 to EBCDIC table was updated
The EBCDIC to ROMAN8 table will be updated too

Input ROMAN8 character to be changed (HEX please):
Do you want to save the changes (Y/N) : Y

Notes:

- * There are two ASCII characters mapping to the same EBCDIC character.
- ** Change the mapping of C8 to its new EBCDIC value.

To display the translation tables, return to the main menu and enter option 7. Then enter the language ID number and the table you want to display.

Modifying 16-Bit Asian ASCII/EBCDIC Translation Tables

To modify the 16-bit Asian ASCII/EBCDIC translation tables for any language other than NATIVE-3000, select option 3 from the LANGINST main menu. When the Modify Native Formats menu is displayed, enter:

1. The language name or language ID number.
 - The modification is aborted if one of the following conditions is met:
 - ☐ Entering a language that is not configured on the system.
 - ☐ Entering NATIVE-3000.
 - ☐ Pressing **Return**.
 - The modification continues if the language requested is configured.
2. If a configured language is entered, the following user dialog is displayed:
 1. Long calendar format
 2. Date format (Calendar format)
 3. Custom date format (Short)
 4. Time format (Clock format)
 5. Currency sign
 6. Currency name
 7. Decimal and thousands separator
 8. Alternate numeric format
 9. YES and NO equivalents
 10. Month names.
 11. Month name abbreviations
 12. Weekday names
 13. Weekday name abbreviations
 14. Direction of text
 15. ASCII/EBCDIC translation tables
 16. Handle truncation in date format
 17. Process the national date table
 18. 16-bit Asian ASCII to EBCDIC translation table
 19. 16-bit Asian EBCDIC to ASCII translation table

Enter selection number : 18

The current default translation value is 0000

Enter the new value (HEX please) : 0000

Input Asian character (HEX please) : c1c4 (*)

Current translation value is (in HEX) : 0000 (*)

Input new translation value (HEX please): a1a1

Input Asian character (HEX please) : a1b2 (*)

Current translation value is (in HEX) : 4FE3 (**)

Input new translation value (HEX please): 4fe3 (***)

Input Asian chracater (HEX please) : 00a5 (****)

*** The character (00a5) is not defined. (****)

Input Asian chracater (HEX please) : RETURN

1. Long calendar format
2. Date format (Calendar format)
3. Custom date format (Short)
4. Time format (Clock format)
5. Currency sign
6. Currency name
7. Decimal and thousands separator
8. Alternate numeric format
9. YES and NO equivalents

. .
. .
. .

Notes:

* If adding a new character, the current translation value is 0000 (in HEX).

** What is currently being mapped.

*** Change to indicated new value.

**** Invalid input was entered.

To display the translation tables, return to the main menu and enter option 7. Then enter the language ID number and the table you want to display.

Table 11-1. LANGINST Error Messages

Message	Cause	Action
A NONNUMERIC GRAPHIC CHARACTER IS EXPECTED ...	An alphabetic or special character (not numeric) is expected.	Enter a valid character.
ATTEMPTING TO ADD TOO MANY CHARACTER SETS.	Adding this language would exceed the maximum configurable character sets.	Do not configure languages from so many character sets.
BUILDING AN EMPTY LANGDEF ...	There was no existing LANGDEF file; a new, empty one is being built.	None. If you have already configured languages, find LANGDEF.PUB.SYS on a backup and restore it; or else, reconfigure the languages with this program.
DELETION TERMINATED ... ATTEMPTING TO DELETE NATIVE-3000.	The language NATIVE-3000 cannot be deleted from the list of configured languages.	None.
ERRONEOUS STARTING YEAR NUMBER. EXPECTED A NUMBER BETWEEN 0 AND 99.	The year number entered is not valid.	Enter the year number again. It must be a number between 0 and 99.
INPUT TOO LONG ... PLEASE REENTER:	The program does not expect this much input in this context.	Re-enter the data correctly.
INTERNAL ERROR ... PLEASE REPORT.	Internal error.	Contact your Hewlett-Packard representative.
INVALID DATE FORMAT. EXPECTED MM/DD/YY.	The entered date is not valid.	Enter the date again in the from MM/DD/YY.
<i>langname</i> IS ALREADY CONFIGURED.	The language selected has already been configured.	None.
<i>langname</i> IS AN ILLEGAL LANGUAGE NAME (OR NUMBER).	The language name or number entered is not valid.	Enter the language again.
<i>langname</i> IS AN INVALID SYSTEM DEFAULT LANGUAGE.	The language selected is not configured on the system.	Add the language to the list of currently configured languages with this program.
<i>langname</i> IS NOT A CONFIGURED LANGUAGE.	The language selected is not configured on the system.	Add the language to the list of currently configured languages with this program.

Table 11-1. LANGINST Error Messages (continued)

Message	Cause	Action
<i>langname</i> IS NOT CONFIGURED.	The language selected is not configured on the system.	Add the language to the list of currently configured languages with this program.
<i>langname</i> IS NOT IN THE CHRDEF FILE.	One of the CHRDEF <i>xx</i> files is not consistent with the NLSDEF file.	Restore all CHRDEF <i>xx</i> files and NLSDEF from your master backup.
NATIVE-3000 IS ALWAYS CONFIGURED.	NATIVE-3000 cannot be added to the list of configured languages; it is always configured.	None.
NATIVE-3000 MAY NOT BE MODIFIED.	The language definition of NATIVE-3000 cannot be modified.	None.
THE CHRDEF <i>xx</i> FILE IS MISSING. THE ADDITION HAS BEEN CANCELLED.	The character definition file for the selected language is missing.	Restore the missing file from your master backup.
THE DECIMAL SEPARATOR AND THOUSANDS SEPARATOR SHOULD BE DIFFERENT.	The decimals and thousands separators have been defined the same.	Change the decimal and/or thousands indicator.
THE EXPECTED NAME SHOULD CONTAIN ALPHABETIC CHARACTERS ONLY.	Only alphabetic characters are allowed in this context.	Please re-enter the value, restricting the input to alphabetic characters.
THE FILECODE FOR CHRDEF <i>xx</i> .PUB.SYS IS INCORRECT.	The character definition file for the selected language has a bad file code.	Restore the missing CHRDEF <i>xx</i> file from the master backup.
THE FILECODE FOR LANGDEF.PUB.SYS IS INCORRECT.	The current language definition file has a bad file code.	Restore LANGDEF.PUB.SYS from a backup copy. Or purge it, and re-create it by reconfiguring the desired languages with this program.
THE FILECODE FOR NLSDEF.PUB.SYS IS INCORRECT.	The master NLS definition file has a bad file code.	Restore NLSDEF.PUB.SYS from the master backup.
THE LANGUAGE YOU ARE ATTEMPTING TO DELETE IS THE SYSTEM DEFAULT LANGUAGE.	The system default language cannot be deleted from the list of configured languages.	To delete this language, first change the system default language to another language.
THE USER SHOULD BE MANAGER.SYS, RUNNING IN THE PUB GROUP.	The user is not MANAGER.PUB.SYS.	Log on as MANAGER.PUB.SYS and run the program again.

Table 11-1. LANGINST Error Messages (continued)

Message	Cause	Action
THERE IS NO MORE ROOM FOR ADDITIONAL DATE PERIODS. PLEASE REPORT.	There is no room for additional entries in the national date table.	Contact your Hewlett-Packard representative.
TOO MANY LANGUAGES HAVE BEEN CONFIGURED.	Adding another language would exceed the maximum configurable languages.	Don't configure so many languages on one system.
UNABLE TO RENAME LANGDEF TO LANGD nnn . THE EXISTING LANGDEF WILL BE PURGED.	The old LANGDEF file cannot be renamed; all files LANGD000 through LANGD999 already exist.	Purge some or all of the files LANGD000 to LANGD999 so the most recent changes to LANGDEF can be saved in the future.
UNKNOWN OPTION.... PLEASE REENTER.	The option selected is not valid.	Enter the number corresponding to one of the valid options.

LINK EDITOR/XL

Use LINK EDITOR/XL to *prepare* native mode (NM) compiled object modules. The LINK EDITOR process resolves all external references which cannot be resolved at run-time. A compiled object module (program file) which has not been *link edited* will not run.

You may also use LINK EDITOR/XL to create and maintain *relocatable* and *executable* libraries.

To prepare and manipulate edit compatibility mode (CM) program files and libraries use the SEGMENTER utility.

Operation

To invoke LINK EDITOR/XL for interactive use enter:

LINKEDIT

LINK EDITOR/XL displays the LinkEd prompt, where you may enter any LINK EDITOR/XL command.

You may also invoke LINK EDITOR/XL with an *info* string that contains a LINK EDITOR/XL command as follows:

RUN LINKEDIT.PUB.SYS;INFO=*info-string*

For example, to link the object modules from the compiled object file LINEDRAW and place the executable module into a file name GRAF, enter the following:

RUN LINKEDIT.PUB.SYS;INFO="LINK LINEDRAW,GRAF"

Commands

The following tables list LINK EDITOR/XL commands in alphabetical order within function.

Table 12-1. Miscellaneous Link Editor/XL Commands

COMMAND	DESCRIPTION
EXIT	Quits an interactive Link Editor session.
LINK	Creates an executable program file.
LISTOBJ	Lists the contents of an object file. The listing shows the symbols contained within the file.
LISTPROG	Lists the contents of an executable program file. The listing shows the symbols contained within the file.

Table 12-2. RL Link Editor Commands

COMMAND	DESCRIPTION
ADDRL	Adds all object modules from a named object file into a relocatable library.
BUILDRL	Builds and initializes a file as a new relocatable library. This library becomes the current relocatable library for subsequent interactive commands.
CLEANRL	Rebuilds a relocatable library by removing any fragmentation and leaving room for 25% expansion within its internal tables.
COPYRL	Copies selected object modules from one relocatable library to another.
EXTRACTRL	Extracts selected object modules from a relocatable library, placing them into a new object file. Modules may be selected by name, locality set, or entry point.
HIDERL	Hides a symbol so the symbol can no longer be used to resolve external references between other modules.
LISTRL	Lists the contents of a relocatable library. The listing shows the names of each object module and the symbols within the library.
PURGERL	Deletes selected object modules from a relocatable library.
REVEALRL	Reveals a symbol that was previously hidden by the HIDERL command.
RL	Selects an existing file as the current relocatable library for subsequent interactive commands.
SHOWRL	Displays the name of the current relocatable library.

Table 12-3. XL Link Editor Commands

COMMAND	DESCRIPTION
ADDXL	Adds all object modules from an object file or a relocatable library to an executable library.
BUILDXL	Builds and initializes a new executable library. This library becomes the current executable library for subsequent interactive commands.
CLEANXL	Rebuilds an executable library by removing any fragmentation and leaving room for 25% expansion within its internal tables.
COPYXL	Copies selected object modules from one executable library to another.
LISTXL	Lists the contents of an executable library. The listing shows the names of each object module and the symbols within the library.
PURGEXL	Deletes selected object modules from an executable library.
SHOWXL	Displays the name of the current executable library.
XL	Selects an existing executable library to be the current executable library for subsequent interactive commands.

Additional Discussion

For more information refer to *HP Link Editor/XL Reference Manual* (32650-90030).

LOGTOOL

The System and Memory Log Analysis Tool (LOGTOOL) lets you display and manage system log files and the memory log file. System log files contain information generated by the operating system. The memory error log file contains memory error information gathered by the memory error logging process MEMLOGP.

You can request that the operating system keep records of certain *users* as well as particular *events*. To keep a certain type of log, change its status to ON using SYSGEN.

To see specific log records, use the LOGTOOL utility (as explained in the “Operation” section, below).

Operation

To invoke LOGTOOL enter:

```
SYSDIAG
DUI > RUN LOGTOOL
```

For detailed information on any command enter HELP followed by the command name. For example, to see information about the LIST command, enter:

```
LOGTOOL> HELP LIST
```

To display data from a set of system log files:

1. Log on as MANAGER.SYS or with SM, OP or DI capability.
2. List the names of log files currently on your system *before* invoking LOGTOOL:

```
LISTFILE LOG@.PUB.SYS
```

3. Invoke LOGTOOL:

```
SYSDIAG
DUI > RUN LOGTOOL
```

4. To get data from your *current* logfile enter the following command to close it and open a new one:

```
LOGTOOL> SWITCHLOG
```

5. If necessary, display logfile record *types*. (You may skip this step if you already are familiar with *types*) :

```
LOGTOOL> TYPES
```

6. Display the analysis for specified logfiles as a formatted list.

```
SYSDIAG>LIST LOG=9/14,17,20,22;TYPE=111,146
```

You may enter the LOG parameter as a *range* of numbers such as 9/14, as a *string* of numbers such as 17,20,22, or as a combination *range* and *string*. (In this example, LOGTOOL will analyze logfiles LOG0009 through LOG0014 and LOG0017 and LOG0020 and LOG0022.)

The TYPE parameter specifies the event types you want analyzed. (In the example, types 111, I/O errors, and type 146, maintenance requests, was specified.)

If you do *not* wish to see the analysis on your terminal screen, but prefer to write the records to an *output* file, you would use the ;OUTFILE parameter as follows:

```
SYSDIAG>LIST LOG=9/14,17,20,22;OUTFILE=MYFILE;TYPE=111,146
```

In this example, the output file MYFILE. You may choose any name so long as it begins with an alphabetic character. LOGTOOL writes the output file to the DIAG group of the SYS account. You may use any HP3000 text editor to examine the output file. You may also copy it with the COPY command or the FCOPY utility.

7. Exit the LOGTOOL utility:

```
LOGTOOL> EXIT  
DUI > EXIT
```

COMMAND SUMMARY

There are three categories of LOGTOOL commands: System Log File Commands (SLF), Memory Log File Commands (MLF), and Miscellaneous Commands (MC). They are briefly described in the following table.

Table 13-1. LOGTOOL Commands

Name	Category	Description
DISPLAYLOG	(SLF)	Displays I/O entries as information is logged.
EXIT	(MC)	Exits LOGTOOL and returns user to DUI.
HELP	(MC)	Gives help on running LOGTOOL.
LAYOUT	(SLF)	Reads in a layout file.
LIST	(SLF)	Lists contents of a system log file.
MEMCLR	(MLF)	Clears the memory logging process log files.
MEMRPT	(MLF)	Displays the contents of the memory log file.
MENTIMER	(MLF)	Alters the timer value of the memory error logging process.
PURGESYSLOG	(SLF)	Deletes the specified system log files from the disc.
PURGEWORK	(SLF)	Deletes the specified work files from the disc.
REDO	(MC)	Edits any of the last four lines of text entered.
SELECT	(SLF)	Selects specified records from the system log files.
STATUS	(SLF)	Reports on the status of all system log files.
SUSPEND	(MC)	Suspends LOGTOOL and returns control to the DUI.
SWITCHLOG	(SLF)	Causes the system to start a new system log file.
TYPES	(SLF)	Describes the system log file “types”.

Logging system events

The following list shows the types of system log events that you can track. To do so, you enable a log event by turning it ON in **SYSGEN**.

SYSGEN System Logging

System Log Events	Event Type
System logging enabled	100
System up record	101
Job initiation record	102
Job termination record	103
Process termination record	104
NM file close record	205
System shutdown record	106
Power failure record	107
Spooling log record	108
I/O error record	111
Physical mount or dismount	112
Logical mount or dismount	113
Tape labels record	114
Console log record	115
Program file event	116
New commercial spooling	120
Architected interface	130
Password changes	134
System logging configuration	135
Restore logging	136
Printer access failure	137
ACD changes	138
Stream initiation logging	139
User logging	140
Process creation	141
Chgroup record	143
File open record	244
Maintenance request log	146
UPS Monitor event logging	148
Diagnostic information record	150
High-priority machine check	152
Low-priority machine check	152
Directory open/close logging	155
CM file close record	160
Chdir	161
Process Adoption	162
File Owner Change	163

All log information is kept in records. Each record begins with a standard header and ends with identification information which varies for each log type. For detailed information about the format of the log records, read *Manager's Guide to MPE/iX Security* (32650-90474).

MKNOD

The **MKNOD** utility lets you create device link files, streams files, and fifo files.

The security restrictions enforced with **MKNOD** are as follows:

- To create a streams file, you must have either SM or NM capability.
- To create a device link file, you must have SM capability.
- To create a fifo device file, you must CD access to the directory in which you're creating the file.

Operation

MKNOD determines how to create the files based on the arguments passed to the program. When creating a device link or streams (character-type) file, you must enter both the major and minor numbers. To create a device link file, enter a major number of 0 and the LDEV number of the device as the minor number. To create a streams file, enter a major number greater than 0.

Syntax

```
mknod "name c|p major minor [link_name]"
```

Parameters

name	Creates a special or fifo file including the program name in the pathname.
c p	Creates a device link file or streams file if this argument is 'c' (character type). Creates a fifo file or pipe if this argument is 'p'.

major	Major number. The following major numbers are valid:
	0 Creates a device link. The minor number represents the LDEV number.
	1-254 Creates a streams file.
minor	Minor device type. This is used in conjunction with the major number option to provide a UNIX compatible device specification. The interpretation of the minor number is dependent on the value of the major number. If the major number is 0, then this number represents a LDEV number. If the major number is > 0, then this number represents a streams connection to a driver.
link_name	Creates the streams file with a link_name . This name is an eight-character MPE link_name (from the NMMGR link screen).

When the special file is created, it can be accessed through the HPFOPEN intrinsic and the C library "open" interface. The O_NONBLOCK option can be specified using open.

MKNOD sets a CI variable **mknodvar** with the value 0 if successful. If errors occur, it returns the file system error status.

To create a fifo file called MYFIFO:

```
mknod "myfifo p"
```

To create a device link file for LDEV 7:

```
mknod "/dev/tape7 c,0,7"
```

To create a streams file with a link_name of DTSLINK, a major number of 1, and a minor number of 10:

```
mknod "strmfile c,1,10,dtslink"
```

You can use the LISTFILE command to display special files. For example:

LISTFILE /dev/@,2

PATH= /dev/./

CODE	SIZE	TYPE	LOGICAL EOF	RECORD LIMIT	SPACE R/B	SECTORS	FILENAME #X MX
	128W	FBf	0	1	1	0	* MYFIFO
	128W	FBs	0	1	1	0	* STRMFILE
	128W	FBd	0	1	1	0	* tape7

:LISTFILE /dev/tape7,5

FILE: /dev/tape7

FILE CODE : 0	FOPTIONS: BINARY, FIXED, NOCCTL, DEVICE LINK
BLK FACTOR: 1	OWNER : **
REC SIZE: 256 (BYTES)	GROUP ID: **
BLK SIZE: 256 (BYTES)	SECURITY--READ :
EXT SIZE: 0 (SECT)	WRITE :
NUM REC: 0	APPEND :
NUM SEC: 0	LOCK :
NUM EXT: 0	EXECUTE:
MAX REC: 1	**SECURITY IS ON
	FLAGS : NO ACCESSORS
NUM LABELS: 0	CREATED : THU, AUG 24, 1993 3:12 PM
MAX LABELS: 0	MODIFIED: THU, AUG 24, 1993 3:12 PM
DISC DEV #: 1	ACCESESED: THU, AUG 24, 1993 3:12 PM
SEC OFFSET: 0	LABEL ADDR: **
VOLNAME : MPEXL_SYSTEM_VOUME_SET: MEMBER1	
DEV TYPE : DEVICE LINK	
LDEV : 7	IO CLASS: TAPE

Additional Discussion

For more information on streams and device link files, refer to the HPDEVCREATE intrinsic in the *MPE/iX Intrinsics Reference Manual* (32650-90028).

N7MF8CNV

N7MF8CNV converts data in EDIT/XL and other MPE text and data files from a Hewlett-Packard 7-bit national substitution character set to ROMAN8.

Operation

N7MF8CNV prompts you for language and file type (text or data). For each data file, you enter the starting position and length of each field (portion of a record) to be converted. For a text file, each record is converted as one field.

The user is prompted for the name of each file to be converted. Files are read one record at a time; each record is converted (or certain fields of it are converted for data files), and the result is written to a new temporary file. When all records have been read, converted, and written to the new file, the old (unconverted) copy is deleted, and the new one saved in its place. An exception to this is KSAM files, which are converted in place, rather than written to a new temporary file.

A count of the number of records read and converted is displayed on \$STDLIST.

This utility will not convert files containing bytes with the eighth bit set. This situation probably indicates a misunderstanding or error. The likely causes are:

- File is not a text or data file.
- File is a data file where the fields have been inaccurately located.
- File was created on a terminal configured for 8-bit operation.
- File has already been converted.

The maximum record length supported is 8192 bytes. The maximum number of fields supported in the records of a data file is 256.

If the file being converted contains user labels, these are copied to the new file without conversion. If a fatal error is encountered during the conversion (for example, 8-bit data or file system error found) the conversion stops, the old copy of the file is saved (with the data unchanged), and the new copy is purged.

An exception to this practise occurs with KSAM files. Since these are converted in place, some records may already have been modified. KSAM files (including key file) should be restored from the backup tape to ensure a consistent copy.

A **CTRL Y** entered during conversion displays the number of records successfully converted, and conversion continues. On variable length data files, if a field or portion of a field is beyond the length of the record just read, a warning is displayed and that field is not converted on that record. Other fields on the same record are converted, and processing continues with subsequent records. After each file has been converted, the user is prompted for another file name.

In addition to the text and data options, there is a test conversion option which shows how the conversion algorithm operates. The test conversion option must be run from a terminal configured for 7-bit operation with the chosen national substitution set. The user is instructed to enter a string, and the result of the conversion is displayed. The user does not have to switch back and forth between 7-bit and 8-bit operation to see the result. Each character converted is displayed as a decimal value in parentheses rather than graphically. Other characters are displayed unchanged.

At any point in the program, a **Return** exits the current program level. A **Return** in response to a request for the starting position and length of a field in a data file indicates that the definition of fields is complete, and the program proceeds with the conversion of the data file. A **Return** entered in response to a request for a text file name indicates the conversion of text files is complete; the program goes back to the question: “Type of file to be converted?”.

NLIOUTIL

The NLIOUTIL utility dynamically activates the Native Language I/O (NLIO) subsystem for Asian and Middle East/African terminals and printers. (NLIO is the basic input and output system integrated into MPE/iX for Native Language Support NLS.) Once activated by NLIOUTIL, properly configured native devices may use the Native Language I/O facility.

Note

You use the System Generator utility (SYSGEN) to configure devices connected directly to the system and the Node Management Configuration Services (NMMGR) to configure devices connected to a Distributed Terminal Controller via Local Area Network.

Operation

NLIOUTIL allows the user to start or stop the NLIO service for the current interactive \$STDIN and \$STDLIST devices. System Supervisor (OP capability) is required to start or stop the NLIO service for a logical device (ldev) or device class name other than the \$STDIN.

When NLIO is started for a device with the NLIOUTIL OPEN command, status information is written to a file named NLIODEF.PUB.SYS. When the system is restarted, NLIO reads in all the previous settings from NLIODEF.PUB.SYS file. There is no need to run NLIOUTIL to open a device unless you wish to change the settings for the device.

The actual update of NLIODEF.PUB.SYS is performed when the NLIOUTIL program is terminated with the END/EXIT/E command.

To invoke NLIOUTIL enter:

NLIOUTIL

or

RUN NLIOUTIL.PUB.SYS;INFO=*infostring*

infostring is an NLIOUTIL command passed with the INFO parameter of the RUN command. NLIOUTIL automatically exits after executing the command passed with the INFO parameter.

NLIOUTIL Commands

The NLIOUTIL utility has the following commands:

HELP	Shows the usage of each NLIOUTIL command.
OPEN	Starts NLIO service for a specified device with a specified language.
SHOW	Displays the status of NLIO service for a specified device.
SHUT	Stops NLIO service for a specified device.
EXIT/END/E	Exits the NLIOUTIL utility.

Each of the commands is explained in the following sections.

HELP Command

The **HELP** command shows the usage of each of the **NLIOUTIL** commands.

```
HELP [HELP | OPEN | SHUT | SHOW | EXIT | END | E ]
```

OPEN Command

The **OPEN** command is used to start the **NLIO** service for a specified device with a specified language.

```
OPEN [nliodevice] [,lang_id] [;RECCONV | ;NORECCONV]
    [;LATIN | ;NONLATIN] [;WIDTH=length]
    [;SHAPE=shape_mode]
```

nliodevice is the ldev or device class name. If the *nliodevice* is omitted, the current stdin/stdlist device is assumed when it is an interactive device. System Supervisor (OP) capability is required to specify devices other than your own stdin.

lang_id is the language number. The supported language numbers are:

- 201 Simplified Chinese
- 211 Traditional Chinese
- 221 Japanese
- 231 Korean
- 51 Arabic

If the *lang_id* is omitted, the user interface language (**USERLANG**) set by **SETJCW** or the system default language is assumed. (The user interface language has the precedence).

Note

You can change the system default language by running the `LANGINST.PUB.SYS` program. Refer to Chapter 11 in this manual for additional information on `LANGINST`.

<code>RECCONV</code>	indicates that a record of the next I/O request that will be made is to be treated as the independent record from the current record. The <code>RECCONV</code> option should be used for terminals. This is the default value used for Asian devices (both terminals and printers).
<code>NORECCONV</code>	indicates that a record of the next I/O request that will be made is to be treated as the continuing data from the current data. The <code>NORECCONV</code> option should be used for printers.
<code>LATIN</code>	specifies the <code>LATIN</code> mode. The primary print direction is left to right. The <code>LATIN</code> mode is used for MEA devices.
<code>NONLATIN</code>	specifies the <code>NONLATIN</code> mode. The primary print direction is right to left. The <code>NONLATIN</code> mode is the default.
<i>length</i>	specifies the width of the print line. If <i>length</i> is not specified, the value is the length specified when the device was configured.
<i>shape_mode</i>	specifies the shape type for MEA characters. The value specified may be 1 for the <i>enhanced</i> font containing 192 shapes, including a number of Lam-Alef ligatures, or 2 for the <i>old</i> font containing 163 shapes, without the Lam-Alef ligatures. The default value is 1.

SHOW Command

The `SHOW` command displays the status of the `NLIO` service for a specified device.

`SHOW` [*nliodevice*]

<i>nliodevice</i>	is the <code>ldev</code> or device class name. If the <i>nliodevice</i> is omitted, the current <code>stdin</code> device is assumed when it is an interactive device. System Supervisor (OP) capability is required to specify devices other than your own <code>stdin</code> . The “at” sign (@) can be specified to designate all devices that are currently opened.
-------------------	---

SHUT Command

The SHUT command stops the NLI0 service for a specified device.

SHUT [*nliodevice*]

nliodevice is the ldev or device class name. If the *nliodevice* is omitted, the current stdin\stdlist device is assumed when it is an interactive device. System Supervisor (OP) capability is required to specify devices other than your own stdin. The “at” sign (@) can be specified to designate all devices that are currently opened.

EXIT/END/E Command

The EXIT, END, or E command exits the NLI0 utility.

EXIT | END | E

Restrictions

- Devices must be configured for NLI0 via SYSGEN or NMMGR.
- System Supervisor (OP) capability is required to specify devices other than your own stdin.
- NLI0 cannot be used for the system console.

NLUTIL

The NLUTIL utility lets you verify the Native Language Support (NLS) Languages and character sets available on the system.

Operation

To display a table showing the currently configured languages and their character set types, enter:

```
NLUTIL
```

or

```
RUN NLUTIL.PUB.SYS
```

NLUTIL displays information in columnar format and then prompts you to request a full listing. For example:

Lang ID	Lang Name	Char ID	Char Name
3	DANISH	1	ROMAN8
5	ENGLISH	1	ROMAN8
12	SPANISH	1	ROMAN8

```
Do you require a full listing of the
current configuration? (Y/N)
```

Enter **Y** to print a full list of the current Native Language Support configuration. Enter **N** to print information about only a single language. For each language set included in the list, NLUTIL reports:

- character attributes
- character collation
- EBCDIC translation tables
- upshift and downshift translation tables
- date and time information
- miscellaneous data on YES/NO indicators
- numeric formats
- monetary symbols

Additional Discussion

For more information refer to *Native Language Programmer's Guide* (32650-90022).

PATCH

The **PATCH** utility accesses, displays, and/or modifies a *compiled, prepped* executable modules. You may make simple changes to program instructions or to global stack area variables with this utility. You may use **PATCH** on compatibility mode programs only.

Before using this utility you should be familiar with machine-executable instructions and the internal format of executable modules in the HP3000 system environment.

Caution

PATCH bypasses normal MPE/iX safeguards and will modify the contents of *privileged* program files. It is therefore possible to corrupt system file(s) or the entire operating system. Hewlett-Packard is *not* responsible for modifications you make to the operating system or system files. For more information contact your Hewlett-Packard service representative.

Operation

To invoke **PATCH** enter:

PATCH

or

RUN PATCH.PUB.SYS

PATCH identifies itself, then displays **ENTITY=?** to request the name of the file you want to list or modify:

```
PROGRAM PATCH G.00.00 (C) HEWLETT-PACKARD CO., 1976
ENTITY =?
```

Enter *filename.groupname.accountname*, in uppercase letters, of the file whose code you wish to display or change. (**PATCH** does not immediately verify that the file you specified is a program file, but you will get an error when you attempt to use a command on a nonprogram file.)

When the file name is accepted, **PATCH** displays the ? prompt, where you enter one of the following four subcommands:

<i>D</i>	Displays code segment contents.
<i>M</i>	Modifies code segment contents.
<i>DG</i>	Displays global area of initial stack.
<i>MG</i>	Modifies global area of initial stack.

After each command, **PATCH** prompts you for another. To terminate **PATCH** press **(Return)** without entering a command. (An incorrectly entered command also terminates **PATCH**.)

On the following pages, each of the four **PATCH** subcommands is explained in detail.

D

The **D** command displays code segment contents. To invoke this command enter:

?D, *segnum*, *address* [, *numlocations*]

Parameters

<i>segnum</i>	The logical segment number to be displayed, found at the top of the PMAP output.
<i>address</i>	The code segment relative address you wish to display is as follows: add the offset of the instruction within the procedure (an octal value found in your compiler listing) to the procedure start address found in the CODE column of your PMAP output.
<i>numlocations</i>	The number of words, in octal, to be displayed.

For a list of segment names and their addresses within a program file use the **PMAP** option of the **PREP** command.

For information regarding the offsets of instructions and global variables in program files, refer to compiler options for the programming language in which the code was written.

M

The **M** command modifies code segment contents. To invoke this command enter:

?M, *segnum*, *address*, [,*numlocations*]

Parameters

<i>segnum</i>	The logical segment number to be modified, found at the top of the PMAP output.
<i>address</i>	The address of the code segment you wish to modify, calculated by adding the offset of the instruction within the procedure (an octal value found in your compiler listing) to the procedure start address found in the CODE column of your PMAP output.
<i>numlocations</i>	The number of words, in octal, to be modified.

When you enter the **M** command, the contents of the current instruction are displayed followed by a comma. To leave the value unchanged, you *must* re-enter the contents! If you press **RETURN** without entering anything the instruction will be set to zeros (%000000, a **NOP** instruction).

For a list of segment names and their addresses within a program file use the **PMAP** option of the **PREP** command.

For information regarding the offsets of instructions and global variables in program files, refer to compiler options for the programming language in which the code was written.

Example

The following example shows you how to use the **M** and **D** commands to display and modify the contents of one instruction in the file **BIGBUCS.PUB.SALES**:

FILE = ? BIGBUCS.PUB.SALES

```
?D,0,20,1
 031042
?M,0,20,1
 031042, 420031
?D,0,20,1
 420031
?
```

DG

The **DG** command displays the global area of the initial stack. To invoke this command enter:

?DG, *reoffset* [, *numwords*]

Parameters

<i>reoffset</i>	The DB-relative offset of the word to display, found in your compiler listing. For more information, refer to the compiler options for the programming language in which the code was written.
<i>numwords</i>	The number of words, in octal, that you wish to display. The default is one.

Example

FILE? PINITRIN.PUB.TEST

?DG,0,4
000010
000015
000000
000046
?

MG

The **MG** command modifies the global area of the initial stack. To invoke this command enter:

?MG, *reoffset* [, *numwords*]

Parameters

<i>reoffset</i>	The DB-relative offset of the word to modify, found in your compiler listing.
<i>numwords</i>	The number of words, in octal, that you wish to modify. The default is one.

The **MG** command displays the contents of the current stack word, followed by a comma. To leave the value unchanged, you *must* re-enter the contents! If you press **(RETURN)** without entering anything the word will be filled with zeros.

For information regarding the offsets of instructions and global variables in program files, refer to compiler options for the programming language in which the code was written.

Examples

Here is an example of using the **MG** and **DG** command. Begin by running **PATCH** and entering the executable file name. In this example, the file is **BIGTECH.PUB.SYS**.

```
PROGRAM PATCH G.00.00 (C) HEWLETT-PACKARD CO., 1976
FILE =?BIGTECH.PUB.SYS
```

The following displays values in the first five addresses:

```
?DG, 0,5
000112
000052
000064
000264
000464
```

The following would set the first five locations to *zero* because **RETURN** is pressed without entering anything after each location is displayed:

```
?MG, 0,5
000112,
000052,
000064,
000264,
000464,
```

The following displays the changes you just made:

```
?DG ,0,5
000000
000000
000000
000000
000000
```

The following changes the zeros displayed above to the values shown after each comma:

```
?MG,0,5
000000,112
000000,52
000000,64
000000,264
000000,464
```

The following displays the first changed value:

```
?DG,,1
-----
000112
```

Additional Discussion

For more information on **PREP** command refer to *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

For more information on User Subprogram Library (USL) files refer to *MPE Segmenter Reference Manual* (30000-90011).

Many compilers produce a listing of global identifiers and their addresses. For more information refer to the reference manual for that compiler's programming language.

PXUTIL

The system manager uses the **PXUTIL** utility to perform operations related to the **UID** and **GID** databases. These include:

- creating the **UID** and **GID** databases
- synchronizing existing **UID** and **GID** database files with the current directory

PXUTIL requires exclusive access to the databases. This means that any command that needs to modify these files (such as **NEWACCT**, **ALTUSER**, and so on) fails during **PXUTIL** operations. For this reason, no activities that access the databases (**NEWACCT**, **ALTACCT**, **PURGEACCT**, **LISTACCT**, **NEWUSER**, **ALTUSER**, **PURGEUSER**, **LISTUSER**, and logon commands) should be attempted on the system while **PXUTIL** is in operation.

Note

If you press **BREAK** and the process is aborted, **MPE/iX** resolves the incomplete operation by closing the existing files (without updating them) and by deleting the new files that it opened.

Operation

You can invoke the **PXUTIL** utility from a session or job. It requires **SM** capability. It is breakable except for the period during which either old database files are being purged or new ones are being saved.

The **PXUTIL** utility supports the following four commands:

- **UPDATE**
- **HELP (H)**
- **QUIT (Q)**
- **EXIT (E)**

Each of these commands is explained on the following pages.

UPDATE

The **UPDATE** command provides a way of ensuring that your UID/GID databases accurately reflect the current directory structure. You may need to update the databases in response to particular error messages generated by various commands that could not properly update the databases as expected. UID/GIDs of existing users and accounts are preserved. If the databases do not exist, they are created from scratch, with UID values beginning at 100 and GID values also beginning at 100.

The UID database contains the following information:

<i>login name</i>	<i>User name.account name.</i> For example, MGR.SYS
<i>uid</i>	Integer initially allocated sequentially for each user.
Initial working directory	Home group. For example, /SYS/PUB.
Initial program to use as a shell	HFS form of the command interpreter (CI .PUB .SYS), which is: /SYS/PUB/CI .

The GID database contains the following information:

group name	Group name. For example, SYS.
<i>gid</i>	Integer initially allocated sequentially for each group.

QUIT

Exits the utility.

EXIT

Exits the utility.

HELP

Displays the **PXUTIL** utility commands.

Examples

The operation from a session is illustrated below:

```
RUN PXUTIL.PUB.SYS
```

```
MPE/iX PXUTIL A.00.01 Copyright (C) Hewlett-Packard 1991. All Rights Reserved.
```

```
PXUTIL> update
```

```
User and group databases sucessfully created.
```

```
PXUTIL> quit
```

Below is a typical invocation and operation of PXUTIL when the databases previously exist. Note that users that were added to the database are displayed. (Refer the commands section for more information on the **UPDATE** command.)

```
MPE/iX PXUTIL A.00.01 Copyright (C) Hewlett-Packard 1991. All Rights Reserved.
```

```
PXUTIL> update
```

```
User  NEWUSER.POSIX with user id= 542 added to user database.
```

```
User  TESTUSER.UI with user id= 543 added to user database.
```

```
User and group databases synchronized with the directory.
```

```
PXUTIL> e
```


SAINT

The Standalone Initialization Utility (**SAINT**) is an interactive utility program that analyzes system libraries (which contain system object modules, or SOMs) to produce executable images known as *boot images*. A bootable image is a file that can be copied directly to memory and executed without modification. The **SAINT** utility's primary function is to produce a boot image for the operating system.

The input file upon which the **SAINT** utility is primarily dependent is the system library file. The format of the library file is defined in the *SOM Architecture Control Document*.

Warning

Do not use this utility without service center support. Unauthorized use will void your warranty and may cause data loss.

The following terms are used in this chapter:

CME	Compatibility mode environment. This file contains the environment required to run a compatibility mode program file created by the MPE segmenter.
IODC	I/O-dependent code. This I/O-device-specific code tests and accesses I/O devices, particularly the boot device (system disk).
IPL	Initial program load. This is the first code to be executed from outside the SPU. It is usually code residing on the system disk (the boot device) and is brought into the system by the PDC and IODC code. It provides a user interface to boot image or SOM files into the system to be executed.
LIF	Logical interchange format. This is a common format used to identify the contents of diverse media.
LST	Library symbol table. This symbol table is contained in SOM library files and defines the meaning and location of symbols contained in the SOMs that are part of the system library file. The library file is defined in the <i>SOM Architecture Control Document</i> .
PIT	Page information table. This table contains entries identical to the page table entries required by the system architecture. These entries describe the physical page layout of the bootable image contained in the boot image file.

PDC	Processor-dependent code. This code is contained in ROM and invokes processor self-tests, locates the boot channel, and checks the condition of the boot device path (including the CPU, main memory, and access to the boot device).
PME	Primary macro environment. The historical name for MPE boot images. This document uses only the term boot image.
RFI	Return from interrupt. This system instruction restores the processor state by resetting the values of the PSW, PC space, and PC offset registers to the values contained in the IPSW and PC space and offset queues. This instruction can be used to switch instruction execution from real addressing mode to virtual addressing mode.
SOM	System object module. The SOM is the file used as the output of system compilers, the input and output of the system linker and MPE/iX loader. Its format is defined in the <i>SOM Architecture Control Document</i> .
XRT	Cross reference table. This is a process-local table built by the MPE/iX loader, which contains information required to make intermodule procedure calls. There is also a system XRT (SXRT) residing in system space that is used to make system calls.
<i>file offset</i>	A byte offset relative to the beginning of the file.
<i>virtual offset</i>	A byte offset relative to the beginning of the space.
<i>space offset</i>	The same as <i>virtual offset</i> .

Preparation for use

To bring up the operating system, the START boot image contains the code that builds the required software structures and initializes the various components of MPE/iX, such as the virtual space manager, memory manager, IPC, compatibility mode emulator, and so forth.

The first code to be executed is in real addressing mode and builds the page table and hash table required to use virtual addressing mode. It then transfers to virtual addressing mode during the call to the virtual entry point.

When the operating system is running, some parts of it are required to be in main memory at all times. This code is part of the system library, and is identified by the memory-resident bit in the library's initialization pointers and subspace dictionary entries. All code that is defined as being memory-resident is contained in the start boot image.

The operating system code, which must reside in memory while the system is being initialized, must also be contained in the boot image. This code is identified by the initially frozen bit contained in the initialization pointers and subspace dictionary entries. All code read is defined as being initially resident and is also contained in the start boot image. During system initialization, the initially frozen code is locked in memory until the system library is fully mapped into virtual space. At that point, the code resets to allow the memory manager to swap it out, if necessary.

Input files

The input files accepted by the **SAINT** utility consist of system library files, CME files, SXRT declaration files, and millicode files. The MPE/iX system library contains MPE/iX system code and data structures. The system millicode is also contained in a SOM, and the compatibility mode definition is contained in a CME file.

The input files required to build the boot image are all identified by one of the load commands (**LOADCME**, **LOADMILLI**, **LOADSXRT**, or **LOADSYSLIB**).

SOMs and system libraries

The system library is the primary input file used by the **SAINT** utility to produce boot images. Libraries contain SOM files, which are the primary output files of system compilers and assemblers, as well as the system linker. Libraries are built by the link editor, which invokes the linker.

In addition to executable code, SOMs contain an initialization pointer area, which is used to define the page allocation and access rights of both code and data. An initialization pointer record defines either the location of code or data within the SOM, or the value and length of a data area required by the SOM. These records are used to allocate pages within the boot image file and to build the page information table, or PIT, for the boot image.

The library symbol table, or LST, contains records defining the location of procedures within the SOMs contained in the library, and these records are copied directly over to the boot image for use by a symbolic debugger when the system is being booted up.

The system object module ACD defines the format of the SOM and the system library.

Operation

Warning

Do not use this utility without service center support. Unauthorized use will void your warranty and may cause data loss.

The boot sequence expected by the boot image is described below:

1. IPL locates the **START** file in the **LIF** directory and reads the first 128 words of that file, which is the operating system boot image.
2. When the file has been identified as a valid bootable image, the file is booted into physical memory.
3. The IPL auxiliary header contained in the boot image identifies the file relative location of the real entry point, and this location is used as an entry point to begin execution of real mode code.
4. Launch first calculates the size of physical memory, and then finds the page information table (PIT) within the boot image to create the system page and hash tables.
5. Next, launch allocates memory for the stack, system globals, and so on and initializes the value of system registers.
6. When launch is ready to switch over to virtual addressing mode, it gets the location of the main virtual entry point from the boot image descriptor, and executes an **RFI** instruction to jump into Genesis.

Boot Image

Use the following SAINT utility commands to creat a boot image:

1. To run the program, type `RUN SAINT.GROUP.ACCT`.
2. Open and read the system library. For example, if your system library file is `NL.ABUILDOO.OFFICIAL`, you enter:

LOADSYSLIB NL.ABUILDOO.OFFICIAL

3. Open and read the CME. For example, if the CME is in the file `CME.ABUILD01.OFFICIAL`, you enter:

LOADME CME.ABUILD01.OFFICIAL

4. Open and read the millicode file. For example, if the millicode file is `EXTMILLI.MILLI.OFFICIAL`, you enter:

LOADMIL EXTMILLI.MILLI.OFFICIAL

5. Load the system cross-reference table and use it to build the boot image. For example, if the SXRT file is `DSXRT.X.X`, you enter:

LOADSXRT DSXRT.X.X
BUILD PME START, SXRT

6. Enter `EXIT` to exit the SAINT utility.

Each of the SAINT utility commands are described below and on the following pages.

BUILD PME

This command creates the permanent boot image from the internal data structures (symbol tables, code arrays, and so on.) built by the SAINT utility in previous load commands. These data structures keep track of the information designated by the load commands previously invoked in this SAINT utility session.

Syntax

`BUILD PME bootimagenam [,SXRT]`

Parameters

<i>bootimage-name</i>	The name of the boot image file. This parameter is required.
<code>SXRT</code>	A keyword designating whether or not a system cross-reference table should be built.

Example

buildpme getit.gotit.good,sxrt

EXIT

This command terminates the **SAINT** utility, closes all files it is currently using, and returns control to the process that invoked this session of the **SAINT** utility.

Syntax

EXIT

Parameters

None

Example

exit

FINDSYM

This command searches for the specified symbol name in the symbol table of the designated boot image and displays pertinent information if the symbol is found. If you do not specify a boot image file, **SAINT** uses the currently built boot image. If a boot image has not yet been built and you do not specify a boot image file, **SAINT** returns an error.

Syntax

FINDSYM *symbolname* [,*filename*]

Parameters

symbolname The symbol to be found in the symbol table. This parameter is required.

filename The name of the boot image file. The default is the currently built boot image.

Example

FINDSYM system_abort,start.abuild01.official

HELP

This command displays a list of the **SAINT** utility commands in alphabetical order, plus a short description of each command's syntax and function. You can also enter the **HELP** command followed by a single command name to see the syntax and function of that specific command.

Syntax

HELP

HELP >*commandname*

Parameters

commandname The name of the command for which you want information.

Examples:

HELP

HELP findsym

LOADCME

This command integrates a compatibility mode environment (CME) file into the current boot image file. **SAINT** appends the CME file to the current boot image and enters the offset of the CME within the boot image in the boot image descriptor record.

Syntax

LOADCME *filename*

Parameters

filename The name of the CME file to be added. This parameter is required.

Example

LOADCME why.captain.spalding

LOADMILLI

This command loads the system millicode file into the boot image.

Syntax

LOADMILLI *filename*

Parameters

filename The name of the SOM file containing the system macros. This parameter is required.

Example

LOADMILLI foobar

LOADSXRT

This command opens the system cross-reference table declaration file (DSXRT), to be used when the SXRT is built by the **BUILDPM**E command. The file is used to build the first section of the SXRT, which continues entries for all system entry points used by switches for CM.

Syntax

```
LOADSXRT filename
```

Parameters

filename The name of the SXRT declaration file. This parameter is required.

Example

```
LOADSXRT dsxrt.loader.exprmnt1
```

LOADSYSLIB

This command opens a system library file and adds the contents to the boot image. It loads only those code pages that are designated as memory resident or initially resident into the boot image; it loads all data pages, regardless of residency declarations.

Syntax

```
LOADSYSLIB filename [realentrypoint]  
                          [virtualentrypoint]  
                          [syslib offset]
```

Parameters

filename The name of the system library file. This parameter is required.

realentrypoint The name of the entry point for the real code in the system library. The default is `init_ivaaddr`.

virtualentrypoint The name of the entry point for the system library entry point. The default is `start`.

syslib offset The virtual offset of the start of the system library. The default value is zero.

Example

```
LOADSYSLIB nl.abuild00.official
```

MAP

This command generates a map of the boot images symbol table. It describes the location of the major components of the boot image. In addition, a flag in the command's parameter list turns on the display of symbolic information obtained from the library symbol tables contained within the boot image.

Syntax

```
MAP [filename[,radix[,sym]]]
```

Parameters

<i>filename</i>	The name of the boot image file used to generate the map and symbol table. Default is PME under construction.
<i>radix</i>	The base in which to display numeric output.
SYM	A keyword which allows the display of symbolic information from the boot images tables.

Examples

MAP

MAP boot imagefile1,H,sym

Error Messages

The following pages list some of the more common error messages you may encounter, the probable cause and a suggested action.

The Physical Destination address is too small, will overlay ISL

CAUSE	The NL used to build this PME has the first-resident or initially resident page at a virtual address that is less than expected. The destination address is the address ISL will use to load the PME. If that destination address is lower than the end of ISL, the PME will not be loaded by ISL.
ACTION	Change the linker commands used to build the NL that is the source of the PME. The linker commands that affect the location of the pages are the limit option on the buildxl command and the cleanxl command.

The Physical Destination address is smaller than expected, MAY overlay ISL

CAUSE	The NL used to build this PME has the first-resident or initially resident page at a virtual address that is less than expected. The destination address is the address that ISL uses to load the PME. If that destination address is lower than the end of ISL, the PME is not loaded by ISL. The destination address may or may not be lower than the end of ISL. The end of ISL is dependent on: ■ type of boot ■ size of machine ■ size of code
ACTION	You may try and use the PME or change the linker commands used to build the NL that is the source of the PME. The linker commands that affect the location of the pages are the limit option on the buildxl command and the cleanxl command.

Internal ERROR. Attempt to read or write with index < 0

CAUSE	SAINT has attempted to read or write a file using an invalid index. This is an error in an input file or an internal error.
ACTION	If an input file is indicated, check the input file for correct format and contents; otherwise enter an SR and provide a copy of all the input files, the SAINT program, and any output file.

An IMPORT STUB entry point was not found in the library

CAUSE	One of the SOMs within the library (the NL loaded by SAINT loadsyslib command) has made a call to a procedure external to itself, and no other SOM in the library contains the procedure.
ACTION	Find out which SOM the called procedure should reside in (that is, console SOM, diagnostic SOM, and so on) and try relinking the library with another version of the SOM.

An SXRT entry point symbol was not found in the library

CAUSE	A procedure or intrinsic name in the DSXRT file (loaded by SAINT loadsxrt command) has no corresponding code in the library (the NL loaded by SAINT loadsyslib command).
ACTION	First make sure that the correct version of the DSXRT file is being used for the library. Once this has been verified, it is most likely that the warning message can be ignored. Sometimes obsolete entries are left in the DSXRT file but have no effect on system operation; however, if a warning message exists <i>and</i> the system is experiencing other problems (for example, will not boot), further diagnosis may be necessary. Contact the factory.

ERROR on move data, file num = xx escape code = yyyyyyy

CAUSE	When SAINT attempts to move data into a file and an error is detected by the operating system a trap code is returned to SAINT . There will be further error messages that will define the problem.
ACTION	Determine the problem from the messages that follow.

Error - OUT OF DISK SPACE or internal pointer error

CAUSE	The most common cause of this error is out of disk space. The other possible cause is that an internal pointer has an invalid value in it.
ACTION	First make sure that there is enough disk space on the volume set where the file is being written. Next check the limits on the group and account where the file is being written. The file name is in the +F-I-L-E—I-N-F-O-R-M-A-T-I-O-N— -D-I-S-P-L-A-Y-+. Correct the disk space problem. If there is enough disk space and the problem still exists, enter an SR describing the problem. Send supporting material: a store tape with a copy of all files used as input to SAINT, (NL,CME, DSXRT,EXTMILLI) a copy of the PME written, and the SAINT program.

Error - Unknown internal pointer error

CAUSE	An internal pointer has an invalid value in it. The file name is in the +F-I-L-E—I-N-F-O-R-M-A-T-I-O-N— -D-I-S-P-L-A-Y+.
ACTION	Enter an SR describing the problem. Send supporting material: A store tape with a copy of all files used as input to SAINT, (NL, CME, DSXRT, EXTMILLI) a copy of the PME written, and the SAINT program.

An SXRT file has not been defined

CAUSE	The buildpme command has been entered with the sxrt option and the loadsxrt file command has not been entered.
ACTION	Enter the “loadsxrt file” command specifying a valid DSXRT file.

Zero SXRT entries found in DSXRT file

CAUSE	The loadsxrt file command was entered specifying an empty DSRT file.
ACTION	Enter the loadsxrt file command specifying a valid DSXRT file.

Internal ERROR. The SXRT file has not been specified

Internal ERROR. Default entries not available

CAUSE SAINT has reached a point where it thinks there is no SXRT file specified and it needs to get SXRT entries. This is an internal error condition.

ACTION Enter an SR describing the problem and the steps taken. Send supporting material: a store tape with a copy of all files used as input to SAINT, (NL, CME, DSXRT, EXTMILLI) a copy of the PME written, and the SAINT program. Re-run the task specifying the `loadsxrt file` command with a valid DSXRT'' file.

Data region found in non-syslib file

CAUSE Wrong file name supplied in `loadmill` command.

ACTION Supply correct file name.

CAUSE Millicode file corrupt.

ACTION Re-install file from tape.

xxx is not a known loader fixup type

CAUSE Millicode or `syslib` file is corrupt.

ACTION Re-install file from tape.

SLPATCH

The SLPATCH utility accesses, displays, and/or modifies segmented library (SL) files.

Before using this utility, you should be familiar with machine-executable instructions and the internal format of segmented library files in the HP3000 system environment.

Caution

SLPATCH bypasses normal MPE/iX safeguards and will modify the contents of *privileged* SL's. It is therefore possible to corrupt the SL or the entire operating system. Hewlett-Packard is *not* responsible for modifications you make to the operating system or system files. For more information contact your Hewlett-Packard service representative.

Operation

To invoke SLPATCH enter:

SLPATCH

or

RUN SLPATCH.PUB.SYS

SLPATCH responds with the following and displays ? to prompt you for the SL file name you wish to work on. Enter the file name in the form: *filename* [*.groupname* [*.acctname*]], and use all uppercase letters. For example:

```
SLPATCH A.43.11 (C) HEWLETT-PACKARD CO., 1976
SL FILE? SL.PUB.HITECH
```

When SLPATCH accepts the name of the segmented library file, it displays another ? to prompt you for a segment name and a command. SLPATCH continues to display this prompt, awaiting another command, until you exit the utility. To terminate SLPATCH, enter EXIT at the ? prompt.

D

This command displays the contents of an SL segment. To invoke this command enter:

? [*segname*,] D, *segdisplace* [, *numwords*]

Parameters

<i>segname</i>	The name of the segment you want to display, obtained from an SLCREF (SL Cross-Reference) listing. This parameter must be specified the first time the segment is accessed, but may be omitted in subsequent commands.
<i>segdisplace</i>	The segment displacement.
<i>numwords</i>	The number of words of the SL to display. The default is one.

M

This command modifies the contents of an SL segment. To invoke this command enter:

? [*segname*,] M, *segdisplace* [, *numwords*]

Parameters

<i>segname</i>	The name of the segment you want to modify, obtained from an SLCREF (SL Cross-Reference) listing. This parameter must be specified the first time the segment is accessed, but may be omitted in subsequent commands.
<i>segdisplace</i>	The segment displacement.
<i>numwords</i>	The number of words SL modifies. The default is one.

The contents of each word to be modified is displayed, followed by a comma. To retain the old value, enter * and press **Return**. To enter a new value, type the number (in octal) and press **Return**. If you only press **Return**, you will be prompted to enter * or a number.

Note

To find the segment displacement, add the instruction offset (generally found in the compiler listing) to the starting address of the procedure (supplied by the **PMAP** option of the **PREP** command).

For information regarding the offsets of instructions in program files, refer to compiler options for the programming language in which the code was written.

SLPATCH Example

Invoke SLPATCH and enter the SL name. For example, if the name of the segmented library file is SL.PUB.HITECH, you enter:

```
SLPATCH
```

```
SLPATCH A.43.11 (C) HEWLETT-PACKARD CO., 1976  
SL FILE? SL.PUB.HITECH
```

Next, display 4 words, beginning with word 0 of segment SDMMCOMM:

```
? SDMMCOMM,D,0,4  
025001  
051404  
041605  
021040
```

Display 3 words, beginning with word 5 in the *same* segment. Notice the segment name need not be entered again:

```
? D,0,3  
023113  
040415  
050641
```

Change the contents of the first word of SDMMCOMM segment and terminate SLPATCH:

```
? M,0,1  
  
025001,025002  
? EXIT
```

Additional Information

For more information on segmented libraries refer to the *MPE Segmenter Reference Manual* (30000-90011).

SOMPATCH

Use the **SOMPATCH** utility for binary modification of a native mode spectrum object module (SOM) program or library file. Binary modification is referred to informally as *patching*.

Warning

Do not use this utility without service center support. Unauthorized use will void your warranty and may cause data loss.

This utility is capable of tracking all of the modifications that it makes to a SOM file. This history is kept in the SOM file in an unloadable buffer space, which affects neither performance nor memory usage of the program. This buffer space is provided by linking in a ypatch file when the executable SOM file is created.

The ypatch file to be linked into SOM 0 of `NL.PUB.SYS` creates loadable code because the storage area needs to be in the boot images, and the **SAINT** utility does not store unloadable spaces in the boot images. This special ypatch file is `ypatsom0` and would normally be used only in the NL.

The **SOMPATCH** utility will accept as input a file not linked with a ypatch file. However, you should use it to build files *with* the ypatch file, so that at any time you can view and query the file and patch history. If the ypatch file is not linked in, no logging is available. All native-mode system products are required to link with the ypatch file if the product is to be supported in the field to any extent with binary patching.

MPE/iX contains a symbolic debugger that can be used for memory patching or to generate patches that you will permanently apply with the **SOMPATCH** utility.

To patch compatibility mode files, use the `slpatch` and `patch` files.

Input Files

The **SOMPATCH** utility has one required file and that is the input file to be queried or patched. This file can be a relocatable or executable library SOM or a program SOM. The **SOMPATCH** utility treats this file as a read-only file. No backup copy is made; however, a **QUIT** command can be used at any time, with this file remaining unchanged.

The user can also optionally specify a second input file, a script file of modification instructions and logging information, referred to hereafter as a *patch file*. A patch file is an ASCII file, created by an editor or other text utility, that specifies one or more commands to the **SOMPATCH** utility. The patch file is given on the command line, or by the **USE** command, to apply one or more patches. Patches can also be applied interactively. Refer to the paragraph “Interactive Patching” under “Operation.”

Error Handling

The input file is maintained in its original state until the user types **EXIT** or **SAVE**. If there is a system malfunction between the **EXIT** command and the colon prompt, purge the possibly modified file and start over with a backup. If no backup is available, redo all patches on the possibly modified input file. (This may cause the oval/verify option on the modify command to generate false error messages). As long as the ypatch file was used, all old patches can be backed out.

Output File

The user can optionally specify an output list file containing all patch instructions issued, any error messages, and any displays. This is saved as a permanent disk file, unless overridden by a user-specified file equation.

The **SOMPATCH** utility writes to the primary file all patches that did not contain errors. If a patch has several modify instructions, or the count is greater than one for one modify, and one or more generated errors, none of the modifies for that patch are applied. Patches are broken up by **LOG** commands.

JCW Handling

The **SOMPATCH** utility sets two JCWs: the system JCW and a private JCW, **PATCHJCW**. **PATCHJCW** is set as follows:

No error -- value of 0
Nonfatal error -- value of 1
Fatal error -- value of 2
Really fatal -- value of 3 (Program will gracefully abort.)

Under these error conditions, the **SOMPATCH** utility also sets the system JCW to the standard MPE values, **OKAY**, **WARN**, and **FATAL**.

Preparation for Use

Patching modifies object code when a problem cannot be easily or conveniently fixed in source code. The **SOMPATCH** utility is used to fix a software problem without recompiling or relinking. A patch is usually developed using the symbolic debugger, with the patch applier using the **SOMPATCH** utility to permanently install a modification in the defective software.

You should link in one of two available ypatch files when you build the product that you are patching. Which one you link depends on how much the product is patched.

When linking a single-SOM library, or a program file, specify the ypatch file you want to use at the end of the source object files for the user's executable SOM. For example:

```
link>LINK from=mysomfile,ypatch4.lib.sys; to=myprogfile
```

When linking a multi-SOM library, specify the desired ypatch file at the end of the source object files for each SOM. For example:

```
link>LINK from=mymod1,mymod2,mymod3,ypatch4.lib.sys; to=myint1
link>LINK from=mymod4,mymod5,mymod6,ypatch4.lib.sys; to=myint2
link>BUILDXL mynl
link>LINK from=myint1,myint2; to=mynl
```

HP product files should contain a minimum of 300 bytes of storage for each anticipated patch. Refer to "Error Handling" for information of how to handle errors when this storage area overflows.

Operation

The MPE/iX command interpreter includes the implied **RUN** concept, which allows the user to invoke a program merely by naming it. For the **SOMPATCH** utility, type **sompatch[.group[.account]]** and then specify a string of parameters (enclosed in quotes) before typing **(Return)**. The program file is searched according to the **CI** variable **HPPATH**. Currently, the **SOMPATCH** utility is stored in **PUB.SYS**.

You may specify the following parameters on the command line:

- name of file to be patched
- name of script file
- name of output list file

The last two files can be specified in one of two ways. To use the C-shell indirection syntax, enter:

```
sompatch mainfile patchfil> listfile
```

Or, to specify the files either positionally or by using the key words `USE` and `LIST`, respectively, enter:

```
sompatch "nl.abuild, patchfil, patchlst"
```

```
sompatch "nl.abuild,,patchlst"
```

```
sompatch "nl.abuild use:patchfil list:patchlst"
```

If you use a patch file as is shown in the example, `SOMPATCH` executes all commands in the patch file and returns you to the MPE prompt.

The following example shows how to patch a file that has been linked with a ypatch file.

Using a text editor, create a patch file that has the following four lines:

```
log junie moon from august skies, 163, 4700741260
; fixes system crash on listf ,5
modify iobuf+20, 2, 12345678 | 22222222 33333333|FFFFFFFF
exit
```

In this command:

Junie moon	is the integrator,
August skies	supplied the patch,
163	is the number of the patch, and
listf ,5	is the system command that needs to be fixed.
;	A semicolon signifies that the rest of the line is a comment.
SR 4700741260	is the STARS SR where the breakdown is documented.
iobuf+20	is the location to modify with an offset of 20 bytes.
2	indicates that two 32-bit words need to be changed.
222222222 and FFFFFFFF	are the new values.
12345678 and 33333333	are the old values.

The file to be patched is `nl.abuild00.official`.

When using a patch file, specify a list file, so that if the JCW is FATAL after the SOMPATCH utility executes, the list file can be examined to determine the problem.

In the following examples, the patch file is pat0511 and the list file is lpat0511.

```
sompatch "nl.abuild00.official <pat0511 >lpat0511"
```

OR

```
sompatch "nl.abuild00.official,use:pat0511,list:lpat0511"
```

Interactive patching

Normally, the LOG command is used with all of the required parameters specified, just as in the example above; however, the SOMPATCH utility also provides user prompting for the log command.

```
: { at colon prompt, type }
sompatch nl.abuild00.official  {invoke program, giving primary file}
sp>log  {a log command is required before the first modify}
username>junie moon from august skies  {program is prompting}
patchid>163 
srnumber>4700241760 
comment>fixes system crash on listf ,5 
comment>  {a  signifies end of comment}
sp>modify iobuf+20, 2
12345678|22222222 33333333|FFFFFFF  {application of the patch}
sp>exit  {exit program, saving patched nl.abuild00.official}

: {the colon prompt returns}
```

; (semicolon)

This command describes the reason that a patch is being applied. It indicates problem symptoms and a fix description. A minimum of one line of comment text is required for each set of modify instructions composing a patch. You should describe the symptoms of the problem in detail, with an explanation when you anticipate a fix in source.

Syntax

; text

Parameters

text The ASCII text describing the problem and why the patch fixes it.

Example

```
;no_op call to check_for_overflow to prevent system error  
;#2099 on bootup.
```

```
;source fix anticipated in SEL CORE release C.06
```

BACKOUT

This command returns a file previously patched to an unmodified state.

It is possible to undo a patch if the patch ID given in the LOG command used to apply the patch is known. This number can be seen in the output of the SHOW [BACKOUTS] command. The patch file name also can be used to back out the set of patches applied with the patch file.

Backout commands are preserved in the ypatch information area. They can be also seen when using the [BACKOUTS] option on the SHOW command.

All backouts are displayed as they are done (at program exit).

Note

This command is not available unless the file being patched has been linked with a ypatch file.

Syntax

```
BA [CKOUT] [ :patchid ] '1b'  
[ @ ] '1b'  
[ file=patchfilename]
```

Parameters

<i>patchid</i>	The patch to be backed out by its identification number (as given with the LOG command).
@	Causes all applied patches to be backed out.
<i>file=</i>	Causes the SOMPATCH utility to look for all patches applied under <i>patchfilename</i> and to back them all out. If there is no reference to <i>patchfilename</i> in the audit history, an error message is issued.
<i>patchfilename</i>	The name of a patch file that was used to make modifications to the primary input file. Group and account default to the logon if not specified.

Example

```
backout :12
```

This example shows 12 as the patch ID specified with the LOG command.

```
backout file=patchfle.pub
```

This example shows that all patches applied under *patchfle.pub* are to be removed.

Error Messages

No patchfile as given.

CAUSE	The FILE= option specified a script file that was not used on this file. The FILES option on the SHOW command indicates what files have been used.
ACTION	Specify the correct file.

No patchid as given.

CAUSE	An attempt was made to back out a patch that the Sompatch utility does not have recorded. The JCWs will be set to their respective FATAL conditions.
ACTION	Use the SHOW command to find out what patches are recorded.

Note

If more than one patch under the same patch ID has been entered, or if patches were applied under the same file name more than once, everything specified is backed out. If in doubt, use the SHOW command to see the buffer's contents for a given patch ID or file name.

DISPLAY

This command displays the value(s) at a location.

Syntax

```
DI[SPLAY] symbolname [+/-offset] [,count] [mode]
```

Parameters

symbolname *Symbolname* is the string name of an external symbol given in the link map for the file being patched.

offset *Offset* is the offset (in bytes) from the start of the symbol. It can be specified as a simple numeric expression. The default value is 0.

count *Count* is the word count (32-bit words) for the command. The default value is 1.

mode *Mode* is the display mode.

The values for *mode* are:

?X Hex (default)

?D Decimal

?O Octal (two 6-digit octal numbers)

?C Characters

The display is sent to \$STDLIST. The parameter *symbolname* may be entered in uppercase or lowercase or mixed case, unless it is defined in a C procedure, in which case it must be entered in the specific case in which it is listed out from the linker. If the symbol starts with an underbar, that should be entered as the first character.

If the values to be displayed are the result of modifications made earlier in the SOMPATCH utility session, then the *oldvalue*|*newvalue* convention of the MODIFY command is used to indicate it. In the example below, only the word at *iobuf+10* has been modified since the user invoked the SOMPATCH utility. (It is currently 1234ABCD in the file and will be 200 if the program is terminated with an EXIT command or a SAVE command.)

```
sp> display iobuf+2,3,?X Return  
22222222 FFFFFFFF 1234ABCD200|
```

Error Messages

Symbol *symbolname* not found.

CAUSE The JCWs are set to their respective WARN status.

ACTION Use the FIND command to search for the symbol.

EXIT

This command is used to exit the program and save the modifications made to the primary input file. To obtain a backup copy, make one before running the **SOMPATCH** utility.

If the **SOMPATCH** utility is being run interactively and patch *xx* had an error, the following message is printed:

```
Error(s) in patch xx but valid modifies will be applied upon
exit.
```

Syntax

EX[IT]

Parameters

None

FIND

This command displays the following information about a symbol:

- residency status (that is, memory resident, initially resident or neither)
- index of SOM (that is, for a multi-SOM library, which SOM contains the symbol)
- symbol type (CODE, DATA, PRIMARY PROGRAM, ...)

This command can also be used to look for a symbol when its name is not known.

Syntax

F[IND] *symbolspec*

Parameters

symbolspec The string name of an external symbol as given in the link map for the file being patched. All symbols are case-sensitive.

The “at” symbol (@) may be used as a leading or trailing mask, as shown in the example below.

Example

The following example displays

```
'trap_handler$248$set_up_user_trap
:find @set_up_user_trap
```

HELP

This command displays a summary of commands, including syntax and options.

Syntax

HE[LP]

Parameters

None

LIST

This command causes the **SOMPATCH** utility to open an output list file.

Syntax

LI[ST] *filename*

Parameters

filename The output list file name. If you omit the group and account, **SOMPATCH** creates the file in the logon group and account.

SOMPATCH uses the output list file to record all commands it encountered, all old values that were modified, and any errors that occurred.

Example

To append output to an existing file, use the C-shell redirection feature. For example:

```
sompatch "mainfile <cmdfile >>listfile"
```

If for some reason the file cannot be opened, an error message is generated and the JCWs are set to WARN status. Reissue the **LIST** command and enter a legal output file name.

LOG

This command records important information about the specified patch.

Syntax

```
LOG [G ]  username ,patchid [,srnumber]
```

Parameters

<i>username</i>	An ASCII string that gives the name of the person installing the patch and the name of the patch creator, if it is not the same person. The string is terminated by a comma, but can include other non-alpha characters.
<i>patchid</i>	The identification code from the patch management system. It is parsed as a string and is terminated by a comma. Blanks are ignored.
<i>srnumber</i>	The SR number that this patch is fixing.

Every patch made on a shared file should have associated with it, at a minimum, the patch ID. The other parameters to this command are also important, but they may not be known at the time that the patch is applied. Always build the file with one of the ypatch files so that this information is saved.

Use the LOG command before the first MODIFY command and before each subsequent patch, where each patch is composed of a set of one or more functionally connected modify commands.

If you do not enter any parameters, LOG will prompt you for them. Refer to the “Operation” section.

Example

```
sp> log STEVE THOMPSON, 155 Return  
sp> log jim gann(by Kate T.), 19, 4700-192066 Return
```

Error Messages

No username given.

CAUSE The first parameter (*username*) is missing.

ACTION Supply *username*.

No patchid given.

CAUSE The second parameter (*patchid*) is missing.

ACTION Supply *patchid*.

File not built with ypatch.

CAUSE The file being patched has not been linked with a
 buffer ypatch file.

ACTION Refer to the “Preparation for Use” information on
 how to do this. JCWs are set to their respective
 FATAL conditions.

MODIFY

This command modifies the value(s) at a specified location.

Syntax

```
MO[DIFY] [symbol] [+/-off] [,count] [?mode] [oval] nval ... '1b'
```

Parameters

<i>symbol</i>	Specifies the name of the associated level 1 symbol. This is normally a CODE type symbol, but SOMPATCH accepts a DATA symbol. If the symbol has duplicates (that is, the same name is used more than once in a multi-SOM library), the SOM command must be used to tell SOMPATCH which SOM to look in for the desired symbol. If the symbol is not specified, absolute addressing from 0 is assumed.
<i>off</i>	The offset (in bytes) from the start of the symbol. It can be specified as a simple numeric expression. The default is 0. If no symbol is specified, the offset must be positive.
<i>count</i>	The word count (32-bit words) for the command. The default is 1. It is not needed for character mode.
<i>mode</i>	Indicates how the values are coming in, which can be one of hex, decimal, character, or octal. The default is hex. Refer to the DISPLAY command for syntax.
<i>oval</i>	The current value of specified location. <i>Oval</i> should be given whenever possible so that SOMPATCH can verify that the modification instruction does indeed reference the correct area in the file. Mode specification is the same as given under the <i>nval</i> entry.
<i>nval</i>	The new value for a specified location. The default base specification mode is hex.

Note

To specify verify on only part of the word, use a “#” character to specify each digit to be ignored.

All modifications are made in a buffer. However, if you issue a **DISPLAY** command following a **MODIFY** command and reference the same location, you will see the *oldvalue|newvalue* pair.

Multi-SOM patches

If a **MODIFY** command belongs to a symbol in a different SOM from the last symbol encountered, and there is no new **LOG** command, and the relevant SOMs have been built with a ypatch file, then the following action is taken:

1. The current patch is written.
2. A new patch is generated, with the same information as the last one (**user**, **patchid**, **srnumber**), but with comments truncated.
3. An informational message is displayed.

Error handling

If there is an error in a **MODIFY** command, the **SOMPATCH** utility takes one of the following actions, depending on its state:

- When running the program interactively, the current **MODIFY** command is not applied, but all previous good **MODIFY** commands are applied.
- If the input to the **SOMPATCH** utility is coming from a patch file, and the SOM being patched has not been built with a ypatch file, then the current **MODIFY** command is not applied, but all previous good **MODIFY** commands are applied.
- If the input to the **SOMPATCH** utility is coming from a patch file, and the SOM being patched has been built with a ypatch file, then none of the current patch (as specified by the last **LOG** command) is applied.

Example

```
sp>modify iobuf+10 2154#### FFFFFFFF Return  
                                     {only check upper 16 bits}
```

Also refer to the “Operation” section for an example of this command.

Error Messages

Symbol *symbolname* not found.

CAUSE PATCHJCW is set to the FATAL condition.

ACTION Change the PATCHJCW setting.

Old value is not as specified.

CAUSE The old value is displayed, and PATCHJCW is set to the FATAL condition.

ACTION Specify the old value.

Illegal syntax—use help.

CAUSE PATCHJCW is set to the FATAL condition.

ACTION Use the help facility, and supply the legal syntax.

OFFSET

This command searches for a set of numbers in a range around a specified symbol.

Syntax

OFFSET *symbolname number0* [*number1 number 2 ... numbern*]

Parameters

symbolname *Symbolname* is an entry from the library symbol table or current SOM dictionary.

numbern *Number0..numbern* is a sequence of numbers that the user is looking for. The maximum is 10 numbers.

This command is used mainly in relocating a patch. When an NL is relinked, all those patches that were not bound to their own entry-level procedure (refer to the **SYMBOL** command) move around. The **OFFSET** command can be used to find their new location.

OPEN

This command causes a patch to open a new file for patching or query. It specifies the input file to be patched. If a file is already open and patches have been made to it, the **SOMPATCH** utility reminds the user that a **SAVE** command is necessary to save changes before opening the new file.

If the file you specified cannot be opened for read/write, an error message is generated, and the JCWs are set to their respective FATAL conditions.

Syntax

OP[EN] *filename*

Parameters

filename The name of the file to be opened.

PATCHFILE

This command causes the **SOMPATCH** utility to generate a patch file from the patches in the main input file. It generates a patch file from the patch information in the ypatch areas in the SOM(s) in the main input file. If there are no patches, a file containing a few lines of header is generated.

If there are no ypatch areas, an error message is generated.

Syntax

PA[TCHFILE] *filename*

Parameters

filename *Filename* is the file to be generated.

QUIT

This command quits the program without changing the input file.

If no modifications were made to the primary input file, the program exits and returns the user to the colon prompt.

If you have made modifications, you will see following prompt:

Do you really want to throw away your patches?

If the user returns an uppercase or lowercase “y” or “yes,” there is an immediate exit to the MPE colon prompt, with no changes made to the file to be patched; otherwise, the **sp>** prompt is displayed, and the **SOMPATCH** utility continues to operate.

In either case, if there is a list file open, it is closed and saved.

Syntax

qu[it]

Parameters

None

SAVE

Use this command to save the modifications made to the primary input file without exiting the program.

The section entitled “Input Files” in the “Overview” section on page 9-1 explains how to recover from a system malfunction in the middle of a **SAVE** command.

Syntax

SA[VE]

Parameters

None

SHOW

Function

This command shows information about any patches that have been applied to the input file.

If only **SHOW** is typed, the most recent patch applied is displayed.

In pre-releases of MPE XL, the installation procedures may not clearly define the proper use of the **CLKUTIL** utility. In this case, the system clock may not be correctly set, and the time of patch shown may be the Greenwich Mean Time, rather than the local time.

Syntax

```
SH[OW] [since[=datespec]
        [= yesterday]
        [:patchid
        [work
        [@
        [srs
        [multi
        [backouts
        [history
        [files
        [symbol+/-off
```

]
]
]
]
]
]
]
]
[,long]

Parameters

since	Specifies a sort backwards by date of the modifications made to the file.
<i>datespec</i>	Specifies the AFTER window to view patch modifications. (The BEFORE window is always the current day). Enter a 1 to 6 digit integer in the form <i>ddmmyy</i> , where either or both <i>mm</i> and <i>yy</i> default to current. If no <i>datespec</i> is given at all, the current day is assumed. (Month starts with January = 01.)
<i>yesterday</i>	A quick way to show all patches since yesterday.
<i>patchid</i>	The string (usually a number) given as the first parameter to the LOG command. The SOMPATCH utility searches its patches to see if that patch has been applied.
work	Shows all patches in the working buffer of an interactive session (that is, the material to be saved by an EXIT command).
@	Provides information on all current patches to be displayed; that is, multiple writes to the same <i>symbol+offset</i> aren't shown. The last write is the only one shown.
srs	Shows all SRs fixed by patches.

multi	Shows all multi-SOM patches.
backouts	Shows all backed-out patches.
history	Shows all modifications made to the symbol specified. It is useful for checking the reworks of a patch.
long	Specifies the long format display. The default is short format, which is patch ID, user name, SR number, and time applied. The long format gives this information, plus the old values, new values, and any other information connected with the patch, such as the name of the source file, the SR number connected with this patch, and the patch applier.
<i>files</i>	Prints out names of all patch files applied and the time that they were applied.
<i>sym- bol+/-offset</i>	Refer to the MODIFY or DISPLAY commands for an explanation of this parameter.

Error Messages

No query available—file not linked with ypatch.

CAUSE	The SOMPATCH utility cannot show anything because the file was not linked with a buffer area for history tracking. PATCHJCW is set to the WARN condition.
ACTION	Link the ypatch.

Illegal date specified—syntax YYMMDD.

CAUSE	The date as specified, with defaults, is not a legal date.
ACTION	Supply a date with the correct syntax.

Symbol not found.

CAUSE	The symbol given in <i>symbolspec</i> was not in the symbol table for the file being queried.
ACTION	Use a correct symbol.

SOM

This command causes the **SOMPATCH** utility to look only in the specified SOM for symbols. There are no SOM errors when parameters are entered as specified. Remember to start indexing from zero.

If a name is given, it is a SOM not in the file. Try **SOM @** to see what SOMs are in the file.

Syntax

```
SOM [somindex ]  
    [somname ]  
    [@       ]  
    [?       ]
```

Parameters

<i>somindex</i>	Specifies an index (starting at 0) of the desired SOM.
<i>somname</i>	Indicates the name of the SOM (as shown by the LISTXL command in linker, or by SOM @.)
@	Shows all SOMs in the current file.
?	Shows which SOM is current for patching.

SYMBOL

This command finds the next level 1 procedure after the symbol+offset specified.

Syntax

```
SYMBOL  symbolname +- offset
```

Parameters

<i>symbolname</i>	The LST symbol name.
<i>offset</i>	The byte offset of the patch from <i>symbolname</i> .

This command is used in generating level 1 patches for multi-SOM libraries.

For a very large file, such as the native mode system library, there may be a five-second to ten-second delay for symbol lookup. In this case, the message **searching ...** is displayed every 15,000 symbols.

For Pascal level 2 procedures, the code being patched is represented as a positive offset from the last level 1 procedure; however, this level 1 procedure is often in a different relocatable object file from the level 2 procedure being patched. This is because of the way that Pascal code is generated:

```

-----
| Proc A's Level 2 code|
|                       |
ProcA + 0000-->-----
|   Proc A -- Level 1   |
|                       |
-----
ProcA + 00xx-->| Proc B's Level 2 code| <-- code being patched
|                       |
ProcB + 0000-->-----
|   Proc B -- Level 1   |
|                       |
-----

```

The patch instruction may have the following syntax, but the second represents the preferred method:

- 1) Modify ProcA + 00xx
- OR
- 2) Modify ProcB - 00yy (Where ProcA + xx + yy = ProcB)

When the multi-SOM NL is linked, the two relocatable object files are sometimes no longer congruent. If they are not congruent, the positive offset from the first level 1 procedure is no longer valid.

There are different ways of dealing with this problem. The easiest way is to specify a patch as a negative offset from the level 1 procedure that owns the level 2 procedure being patched (2) in the example above). The **SOMPATCH** utility contains the **SYMBOL** command, which determines the procedure name of the level 1 procedure that owns the level 2 procedure containing the patched instruction, plus the negative offset of the patch from the level 1 procedure entry point.

Error Message

No such symbol

CAUSE	A symbol was given that is not in the library symbol table.
ACTION	Try the FIND command, and use the @ feature.

USE

This command causes a patch to read an input file of patch instructions, and to implement the instructions.

If there is an error such as an undefined symbol in a **MODIFY** command, an error message is generated, and the program continues to execute.

If the file specified is not present, an error message is generated, and the JCWs are set to their respective FATAL status.

When the end of the patch file is reached, the program returns to the MPE/iX CI.

Syntax

US[E] *filename*

Parameters

filename *Filename* is the name of the script file to be used.
The group and account default to logon if not specified.

VERSION

This command gives the current **SOMPATCH** utility version. It is used to show the version of the **SOMPATCH** utility that is being run. The version is also printed on the list file and as part of each patch written to the ypatch area.

Syntax

VE[RSION]

Parameters

None

Example

```
sp> VERSION Return  
Sompatch Version A.00.00  
sp>
```

SORT-MERGE/XL

SORT-MERGE/XL allows you to sort files by single or multiple key fields and to merge two or more sorted files into a single new sorted file.

You may use SORT-MERGE/XL as a standalone utility (either interactively or in batch mode), or from within a program. For information on how to use SORT-MERGE/XL programmatically, refer to *SORT-MERGE/XL Programmer's Guide* (32650-90080).

Operation

Using SORT

To invoke the *sort* portion of SORT-MERGE/XL enter:

SORT

or

RUN SORT.PUB.SYS

SORT-MERGE/XL responds with a message similar to this:

```
HP31900A.01.01  SORT/V    TUE, JAN  2, 1990,  2:58 PM
(C) HEWLETT-PACKARD CO. 1987
```

It then presents a > prompt. At this prompt you must specify an INPUT file, an OUTPUT file and sort KEY(s). For example:

```
> INPUT INFILE
> OUTPUT OUTFILE
> KEY 1,10
> KEY 11,6
> END
```

You may enter the parameters in any order. However, you must specify the keywords OUTPUT, INPUT and KEY. You must use valid MPE/iX file names. If you qualify both files with group and account names, you must have *read* access to the INPUT file and *write* access to the group and account for the OUTPUT file.

This example contains two keys, the first, beginning in byte one, which is 10 bytes long and the second, beginning in byte 11, which

is six bytes long. The first key entered is the *primary* key. Each successive key entered is *secondary* to the key entered before it.

The word **END** indicates you are finished entering parameters and signals **SORT-MERGE/XL** to begin processing.

After sorting your file **SORT-MERGE/XL** produces a statistical display and then terminates.

If you want to terminate **SORT-MERGE/XL** *without* processing do *not* enter **END**. Instead enter:

```
> EXIT
```

You may access the sorted **OUTPUT** file just as you would any other **MPE/iX** file.

Using **MERGE**

To invoke the **MERGE** portion of **SORT-MERGE/XL** enter:

```
MERGE
```

or

```
RUN MERGE.PUB.SYS
```

SORT-MERGE/XL responds with a display similar to:

```
HP31900A.01.01  MERGE/V      TUE, JAN  2, 1990,  3:54 PM
(C) HEWLETT-PACKARD CO. 1987
```

It then presents a **>** prompt. At this prompt you must supply **INPUT** file(s) an **OUTPUT** file and **SORT** key(s).

For example:

```
> INPUT INFILE1, INFILE2, ... INFILEN
> OUTPUT OUTFILE
> KEY 1,12
> END
```

You may enter the parameters in any order. However, you must specify the keywords **INPUT**, **OUTPUT** and **KEY**. You may enter file names on one line separated by commas (as in the example), or you may enter each on a separate line with the keyword **INPUT**.

The **OUTPUT** file is the file to which the merged files will be written. In this example there is only one key. It begins in byte one and is 12 bytes long.

The word **END** indicates you are finished entering parameters and signals **SORT-MERGE/XL** to begin processing.

After merging your files **SORT-MERGE/XL** produces a statistical display and then terminates.

If you want to terminate **SORT-MERGE/XL** *without* processing do not enter **END**. Instead enter:

EXIT

You may access the sorted OUTPUT file just as you would any other MPE/iX file.

Additional Discussion

For more information refer to *SORT-MERGE/XL General User's Guide* (32650-90082).

SPIFF

Use the Native Mode Spool File Interface Facility to list, manipulate, and transfer spooled device files (spoolfiles) that are created and maintained by MPE/iX. **SPIFF** replaces the CM **SPOOK** program.

SPIFF supports many of the commands of the **SPOOK** program, and has also been enhanced with the following features:

- Access to the MPE/iX Command Interpreter.
- Case insensitive **FIND**, which you may set as the default.
- Native mode output display.
- No **LOCKED** state for spool files. You can text in a file or output it to tape without changing its state.
- New **STORE** and **BROWSE** commands.
- Allows use of file equations on **\$STDINX** (the standard input file) and **\$STDLIST** (the standard output file).

Operation

To run the Native Mode Spooler Utility, enter **SPIFF**. You will see an identifying banner and the **>** prompt, like this:

```
SPIFF A.00.00 (C) COPYRIGHT HEWLETT-PACKARD CO. 1993
```

```
>
```

Once you see the prompt, you may enter any of the commands described on the following page.

Table 24-1. SPIFF Commands

COMMAND	DESCRIPTION
ALTER	Alters the priority, number of copies, target device, or any combination of these attributes, of one spoolfile or many spoolfiles.
APPEND	Appends all or part of one or many spoolfiles to a new spoolfile. The first spoolfile processed by the command creates the new spoolfile. Subsequent spoolfiles are appended to it.
BROWSE	Invokes the HPBROWSE utility, if it is available.
COPY	Copies all or part of one or many spoolfiles to a new spoolfile.
DEBUG	Invokes the MPE/iX DEBUG facility if the SPIFF user has Privileged Mode (PM) capability.
EXIT	Terminates SPIFF , returning control to its parent process.
FIND	Locates a specified pattern in a specified range of the current spoolfile.
HELP	Displays information about SPIFF and its commands.
INPUT	Inputs one or more spoolfiles from a tape created by SPOOK5 or SPFXFER .
LIST	Lists a line range of the currently TEXT ed spoolfile to \$STDLIST.
MODE	Controls the width and format of the displayed output of the LIST and FIND commands.
OUTPUT	Outputs one or more spoolfiles to a tape in SPOOK5/SPFXFER format.
PURGE	Deletes one or more spoolfiles from the system.
QUIT	Terminates SPIFF , returning control to its parent process.
SHOW	Displays information about one or more spoolfiles.
STORE	Stores one or more files to tape using the MPE/iX STORE subsystem.
TEXT	Accesses an output spoolfile for use by the ALTER , APPEND , BROWSE , COPY , FIND , LIST , PURGE , and SHOW commands.
XPLAIN	Displays a summary of SPIFF commands.

SPIFF recognizes only the first letter of the full form of the command (**F** or **FIND** in the above example), except for **APPEND** and **STORE**. The abbreviations for these two commands require two letters (**AP**, **ST**) to distinguish them from the **ALTER** (**A**) and **SHOW** (**S**) abbreviations, respectively.

File Definitions

SPIFF opens the formal file designator **SPUTIN** as its **\$STDIN(X)** and the formal file designator **SPUTOUT** as its **\$STDLIST**. You may redirect these files as desired with a file equation. However the record width of any redirected **SPUTOUT** should not be less than 80 bytes; otherwise displays and messages may generate an error when **SPIFF** directs them to **SPUTOUT**.

Additional Discussion

The *Native Mode Spooler Reference Manual* (32650-90166) presents a detailed description of the **SPIFF** utility and its commands.

STANDARDS

The system bootstrap, initial program load (IPL) and initial system load (ISL) standard provides a standard interface through which any *Hewlett-Packard Precision Architecture* (PA-RISC) computer can boot any operating system. The standard also provides a common user interface for booting PA-RISC systems.

Warning

**Do not use this information without service center support.
Unauthorized use will void your warranty and may cause data loss.**

The following terms are used in this chapter:

PDC	Refers to processor-dependent code as defined in the <i>System I/O Architectural Control Document</i> (ACD).
IODC	I/O-dependent code is I/O-device-specific code used to test and access I/O devices (also defined in the I/O ACD).
BOOTSTRAP	BOOTSTRAP describes the PDC and IODC functionality needed to bring code into memory and launch it.
IPL	Initial program load is the first code brought into the system and executed from outside the SPU. This code is loaded by the bootstrap code.
ISL	Initial system load is a standard code module that is used during the startup of any operating system to provide a standard user interface for booting. On some systems, this code may be the IPL, while on others, IPL may perform some preliminary tasks and then proceed to load the ISL.

The bootstrap and initial system load are discussed on the following pages.

The bootstrap

The bootstrap performs these functions:

- a minimum SPU test including memory configuration, nondestructive memory test, and destructive SPU test and one coldload device path
- provides user interface with system initialization information and alters the initialization path.
- reads in the IPL code from the one load device, finds the location of the IPL on the device through a pointer in the device label, and loads and launches the IPL code.

The initial system loader

The Initial System Loader performs these functions:

- provides user interface with boot path information and alters boot path.
- loads an operating system-specific code set or a hardware-specific code set and launches it. If this implementation-specific code does not damage the ISL image, ISL remains in memory in case the code returns control to ISL for initialization of further utilities.

Certain standards are used by hardware and software operating systems using system bootstrap, IPL, and ISL standard. These standards are architectural in nature, but are not necessarily defined in any system architectural document. The remainder of this chapter contains a description of these standards.

Stable storage and nonvolatile memory layout

Stable storage and nonvolatile memory (NVM) are described as blocks of bytes that are accessible by bootstrap, ISL, and the operating system through standard entry points. The first 96 bytes of stable storage are required implementation; bytes 96 through 191 are optional, but if implemented they are reserved for PDC and ISL access as described below. Nonvolatile memory is not required by the architecture; therefore, it should contain only values that can be managed by an alternate method. When more than one byte is used in the representation of an item in stable storage, the most significant byte is the byte with the lower address.

The format of stable storage is shown below:

Table 25-1. Stable storage format

BYTE	CONTENTS
0-31	Boot flags and device
32-63	Unique file names
64-95	Future OS requirements
96-127	Console terminal
128-159	Alternate boot path
160-191	Dump flags and device
192- <i>nnn</i>	Future OS options

Autoboot flags and device path

This 32-byte field defines the coldload path that PDC uses for autoboot. If this path is not valid, or if the device that it describes is not a valid boot device containing an IPL image, PDC then requests a valid path through the console if the console is present; otherwise, the error is displayed through the front panel.

A detailed internal representation of a boot or console path (the format applies to the auto bootpath, the alternate boot path, the dump path, and the console path) is as follows:

Table 25-2. Boot or console path

0000	flags	BC(0)	BC(1)	BC(2)
0004	BC(3)	BC(4)	BC(5)	MOD
0008		Logical_ID		
000C		Device_Depend		
001F				
0020				

Note that in the above illustration, the flags field in the console path is ignored. The format of flags is as follows:

Format of flags

Table 25-3. Format of flags

0	1	2-3	4 through 7
ab	as	00	timer

Console Path

This 32-byte field defines the device path that PDC uses to locate the system console. If this path is not valid, or if the device that it describes is not a valid console device image, PDC then uses a default path.

Alternate Boot Path

This 32-byte field defines the coldload path that PDC uses (after getting the go-ahead from the operator) if the operator rejects the autoboot path through console intervention. If this path is not valid, or if the device it describes is not a valid boot device containing an IPL image, PDC then requests a valid path through the console.

Dump flags and device

This 32-byte field is used to describe the destination device for a snapshot dump facility.

The format of nonvolatile memory is shown below:

Table 25-4. Nonvolatile memory

BYTE	CONTENTS
0-63	PDC and boot reserved
64-nn	OS reserved

NVM is an optional implementation, not required by the architecture. If NVM is implemented, the PDC and boot reserved area of NVM is as follows:

Table 25-5. PDC and boot area for NVM

BYTE	CONTENTS
0-1F	Last boot-device path
20-23	Self-test status
24-27	Powerfail time stamp
28-2A	Boot restart time stamp
2B-2F	TOC restart time stamp
30-63	PDC and boot reserved

LIF standard

The Hewlett-Packard Logical Interchange Format (LIF) provides a standard method for locating IPL code on the boot device.

The method of locating IPL code on the boot device must be the same for all boot devices, computer processors, and the operating systems, so that the PDC knows where to find it. Having IPL at the very beginning of the device is ideal; however, some operating systems require a volume label at the beginning of a disk. IPL also needs a method of locating its modules (utilities) when they are needed.

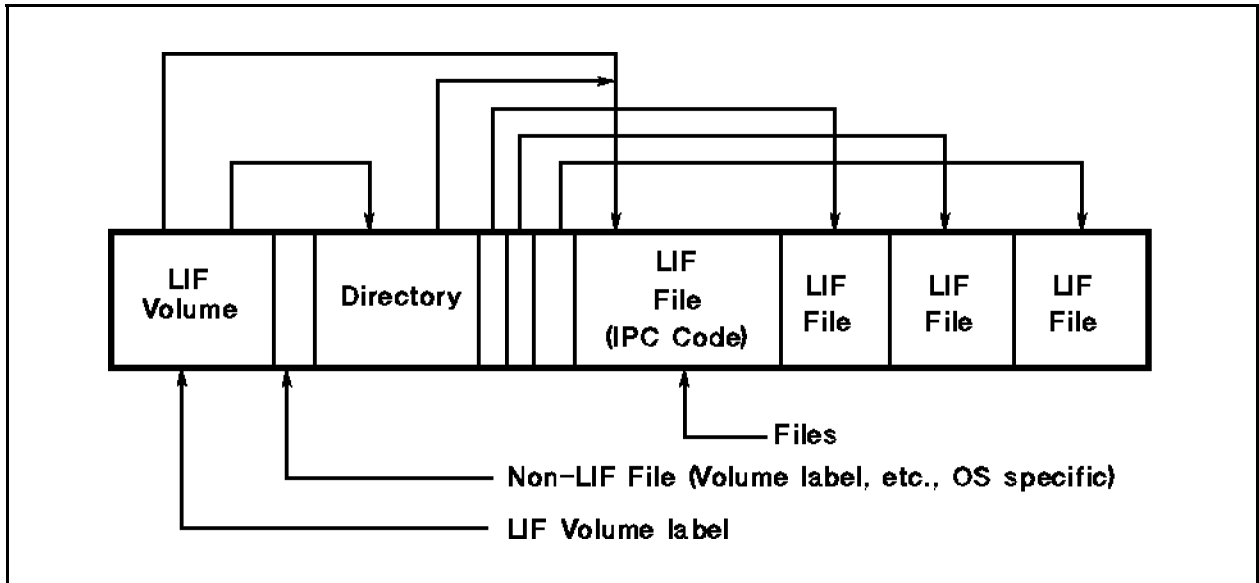
The LIF standard addresses these issues, and provides a standard with utilities already in existence. For further information on LIF format, refer to LIF Directory Organization and Record Format for Data Interchange.

LIF requires a 256-byte volume label at the beginning of the media; thus, the operating-system-specific volume label can be located at the beginning of the disk, offset by 256 bytes. The LIF volume label points to the location of the LIF directory, which then points to the location of each of the files.

Compliance to LIF standard does not require complete implementation. The level of compliance to the standard in the IPL code is the minimum implementation (volume header and directory in ASCII code). The format of the files is the system object module (SOM) format.

The LIF volume header, as well as an entry in the LIF directory, points to the IPL code.

The following drawing represents the LIF standard logical layout of disk and tape media.



LIF Standard logical layout

The LIF volume label allows easy identification of media type and gives the location of the directory. The format of the LIF volume label follows:

Table 25-6. LIF volume label format

BYTE	CONTENTS
0-1	LIF identifier
2-7	Volume label (0-6 ASCII characters)
8-11	Directory start address (in blocks)
12-13	Octal 10000
14-15	Set to 0 (dummy)
16-19	Length of directory (fixed at initialization)
20-21	Set to 0
22-23	Set to 0
24-41	Set to 0 (level 1 extension)
42-239	Set to 0 (reserved for extensions and future use)
240-243	Byte address of IPL on media
244-247	IPL length (in bytes)
248-251	Offset in IPL of entry
252-253	Set to 0
254-255	Set to 0 (reserved by system 250)

The directory contains all of the information necessary to find files. It is a linear list of 32-byte directory entries, one for each LIF file on the media. The maximum number of entries in the directory is fixed at the time of initialization. A logical end of directory mark is defined to be a file type of -1 and is written only if the directory is not filled. The physical end of directory is determined by adding the start of directory and length of directory fields from the volume label. Directory entries must be stored so that they are in strictly increasing starting addresses on sequential media. Directory entries are undefined after the logical end of directory, so when a file is appended to the directory, the following directory entry's file type must be set to -1 to make it the logical end of the directory.

LIF addressing is in blocks of 256 bytes, and system addresses are 2K-bytes-aligned.

Each directory is organized as follows:

Table 25-7. LIF addressing

BYTE	CONTENTS
0-9	File name (1-10 ASCII characters, trailing blanks)
10-11	File type
12-15	Starting address (in blocks)
16-19	Length of file (in blocks)
20-25	Time of creation
26-27	1 /volume number
28-31	Set to 0 (implementation)

The file type is a 16-bit signed integer. The defined file types that are recognized by systems are

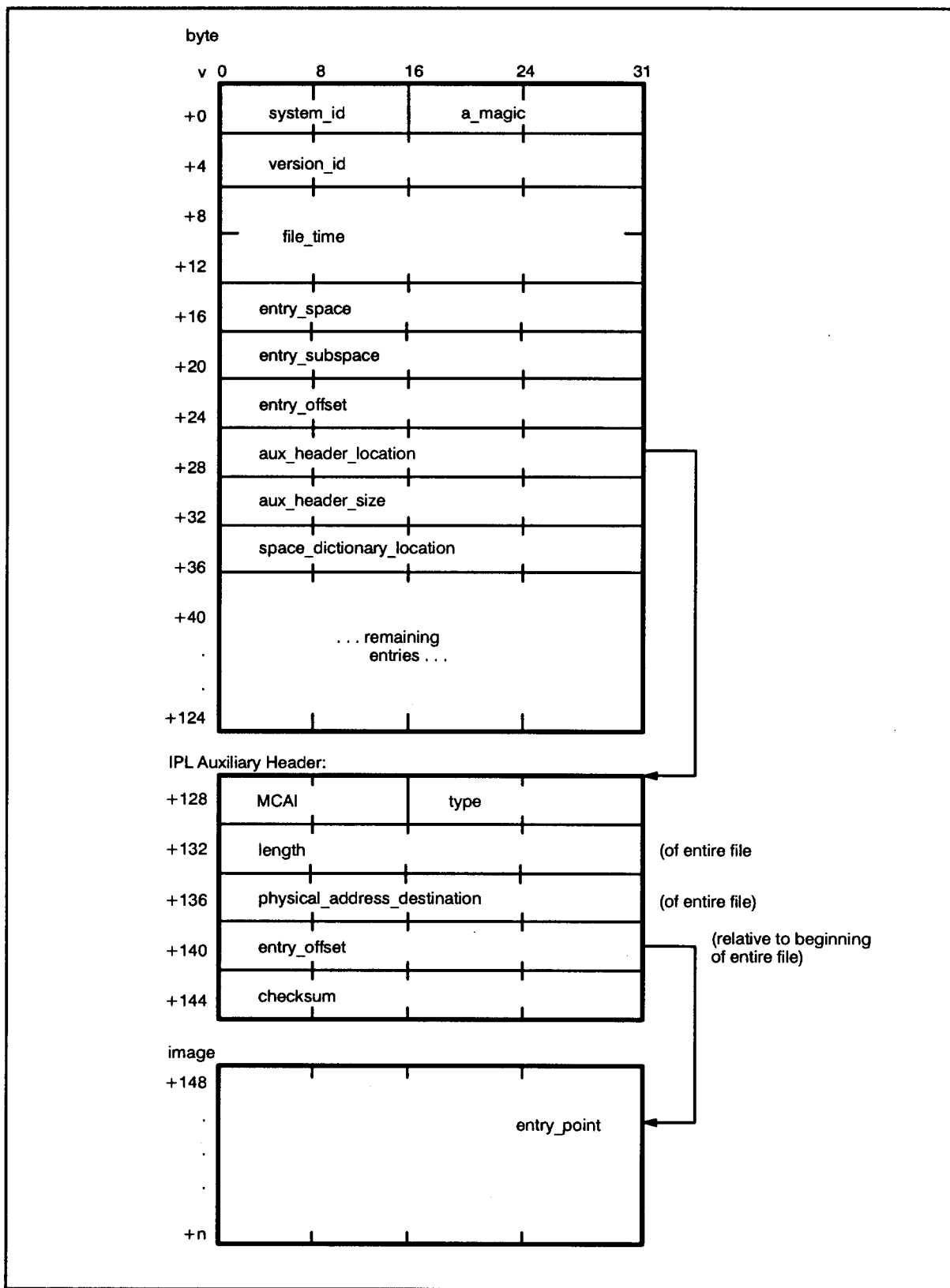
- 0—Purged file
- 30001—Bootable, executable file
- 30002—Boot data file
- 30003—Autoexecutable file list
- 30004—Data protect file
- 30010—HPE system file

A file is deleted from the directory by changing its file type to -2, to represent a purged file. The data itself need not be removed from the media.

Bootable utility format

All of the bootable utilities accessible through the LIF directory must have enough of a common format for IPL to load and launch the utilities through a standard method. IPL may need to know the intended physical memory destination address for which the module was linked, as well as the length of the image and the entry point. For those utilities that are position independent, the destination address can be set to -1 and IPL will load it at the first available memory after IPL.

All software implementation intends to support the system object module (SOM) format, using the linker; therefore, an auxiliary SOM header for IPL, as described below, would meet IPL's needs for loading and launching bootable utilities. For further description of the linker and the SOM format, refer to the *System Linker External Specifications* and the *System Object File Format Architectural Control Document*.



LG200070_002

Figure 25-1. Boot utility format

Main Memory Layout

Although the exact memory locations of boot and IPL code during the boot process vary according to size of the IODC and IPL, a general description of memory layout is presented here for clarification. Also, the first page of main memory is reserved for communication between PDC and software.

Table 25-8. Main memory layout

X'00000000	Initialize vectors	0
X'00000040	Processor dependent	64
X'00000200	Reserved	512
X'00000350	Memory configuration	848
X'00000360	MEM_ERR	864
X'00000380	MEM_FREE	896
X'00000384	MEM_HPA	900
X'00000388	MEM_PDC	904
X'0000038C	MEM_10MSEC	908
X'00000390	Initial memory module	912
X'000003A0	Boot console/display	928
X'000003D0	Boot device	976
X'00000400	Boot keyboard	1024
X'00000430	Reserved	1072
X'00000600	Processor dependent	1536
X'00000800		2048

The format of the first memory controller configuration is as follows:

- word0: HPA of the memory controller
- word1: SPA of the corresponding memory
- word2: SPA_size
- word3: Max_Mem

The format of the console terminal and boot device configurations are as follows:

Table 25-9.
Console terminal and boot device configuration

X'00	path	00
X'08	LAYERS	08
X'20	HPA	322
X'24	SPA	36
X'28	IODC_IO	40
X'2C	Reserved	44
X'30	Class	48

The format of the boot and system console device paths is the same as that for the autoboot, alternate boot, and console paths in stable storage. In the console path, the flags are ignored.

The offsets of IODC, IPL, and booted utilities are variable, based on the size of code images.

Table 25-10. IODC, IPL, and booted utilities offsets

X'00000000	Page zero	0
X'00000800		2048
MEM_FREE	Monarch PDC	MEM_FREE
	Console IODC	
	Boot Device IODC	
IPL_START	IPL code	IPL_START

SYSMAP

SYSMAP maps the hardware configuration of I/O devices, CPU boards and memory boards.

Operation

To invoke SYSMAP enter:

```
SYSDIAG  
DUI> RUN SYSMAP
```

SYSMAP is part of the Online Diagnostics Subsystem. To use this utility, start the diagnostic subsystem by entering SYSDIAG and at the DUI> prompt, enter RUN SYSMAP.

Once you are in the SYSMAP subsystem, you can choose one of three basic map options:

```
ENTER MAP> CPUMAP  
ENTER MAP> MEMMAP  
ENTER MAP> IOMAP
```

If you choose CPUMAP the system displays the following for each CPU: hardware model and revision number, software model and revision number and slot number.

If you choose MEMMAP the system responds by listing memory controller's and arrays, slot numbers, hard physical addresses and memory size.

If you choose IOMAP the system responds by presenting the IOMAP> prompt. At this prompt you may choose one of three responses:

```
IOMAP> GENERAL  
IOMAP> CLASS  
IOMAP> STEPPING
```

- **GENERAL** displays a general map of the I/O configuration.
- **CLASS** maps all devices in the device class you specify.
- **STEPPING** displays all system hardware configurations or all configurations from a specific physical address on down.

Additional Discussion

*Precision Architecture: HP 3000/9xx & HP 9000/8xx Online
Diagnostics Subsystem Utilities Manual (09740-64007).*

tic

The **tic** utility compiles the **terminfo** source file.

Operation

The **tic** utility compiles source **terminfo** descriptions. The compiled entry is installed under the **/usr/lib/terminfo** directory hierarchy. If the **TERMINFO** environment variable is set, results are placed in the directory it points to instead. Entries are stored in directories that match the first character of their name. The entry for the VT-100 terminal, for example, is stored in **/usr/lib/terminfo/v/vt100**.

When a *use=* specification is found in the source file, **tic** adds the binary of the appropriate capabilities to the compiled file.

The **tic** utility can only be used to compile single files. Multiple files can be compiled by concatenating individual files together.

Compiled entries cannot exceed 4098 bytes. The name field cannot exceed 128 bytes. Terminal names that are longer than 14 characters are truncated to 14 characters, and a warning message is printed.

Syntax

```
tic.hpbin.sys [ -v[ n ] ] -c file
```

Parameters

-v[<i>n</i>]	provides varying level of detail on the compilation process where <i>n</i> is a number from 1 to 10 (default is 1). The higher the value for <i>n</i> , the more detail provided.
-c	checks for errors in file (not including errors in <i>use=links</i>).

Example

```
tic.hpbin.sys /product/curses/lib/terminfo/ansi
```

Related Information

Commands **untic** utility, **terminfo** database

TERMDSM

TERMDSM is part of the Online Diagnostics Subsystem and is used to diagnose, dump, and reset logical devices, ports, and distributed terminal controllers. It also performs status checks of ports and DTC's.

Operation

To invoke TERMDSM enter:

```
SYSDIAG
DUI> RUN TERMDSM
```

At this point the system responds by displaying the **TERMDSM** banner, your system's Distributed Terminal Controllers and the following prompt:

```
Comment  DIag  DTc  DUmp  Help  Reset  Status  EXIt ?
```

To proceed, select one of these eight commands by entering either the entire command or only the letters shown in upper case.

Each **TERMDSM** command has a security level which prevents users without certain capabilities from executing it. The following shows which capabilities are required for each of three **TERMDSM** security levels.

User Capability	Security Level
SM or DI	0 (greatest capability)
SM, DI or OP	1 (second most capability)
SM, DI, OP or AM	2 (least capability)

Each of the **TERMDSM** commands is explained on the following pages.

Comment

Comment allows comments to be entered for later reference. Security level is 2.

Diag

DIag is entered to run diagnostic functions. The system responds by displaying the following:

(diagnostics)

SElftest # (DTC Selftest)

PRint #,#,# (Print to specified port)

Internal #,#,# (Internal loopback on specified port)

EXTernal #,#,# (Loopback through hood on specified port)

Terminal #,#,# (write and read to terminal on that port)

(carriage return to exit) ?

The parameter for SElftest is DTC number. Parameters for all other tests consist of DTC number, SIC number and port number. For example, to execute a PRint test on port 7 on SIC 0 on DTC 2 the following would be entered: PR 2,0,7.

Security level is 0 for DTC selftest; 1 for all other diagnostic tests.

Caution

A DTC SElftest aborts all sessions on the DTC being tested and may result in lost data.

DUmp

DUmp dumps the contents of memory for the port and driver to disk. When **DU** is entered the system responds by displaying the following:

(dump)

Ldev #

POrt #,#,#

DTc #

(carriage return to exit) ?

Parameters are Ldev number or DTc number. For POrt the parameters are DTC, SIC and port number. Security level is 0 for DTC dump, 1 for other functions. Ports and Ldevs are dumped to a file with the following naming convention: TRMnnnXX.pub.sys where nnn is the day of year and XX is two characters ranging from AA to ZZ. DTC's are dumped to a file named as follows: SnnnnnnX.pub.sys, where nnnnnn is the 6-digit DTC address and X ranges from A to Z. Analyzing dump contents requires specialized training and is usually done by HP support personnel.

Reset

Reset resets a DTC, SIC or port. When the **Reset** command is entered the system responds with the following:

(reset)

Ldev #

POrt #,#,#

SIC #,# (multiplexer card in DTC)

DTc #

(carriage return to exit) ?

Parameters are DTC, SIC and port for **POrt** and DTC and SIC for **SIC**. Other parameters are as indicated. Security is 1 for Ldev's and ports; 0 for DTC's or SIC's.

Caution

Active sessions are logged off reset devices.

Status

Status displays information about the DTC or port. When **Status** is entered the system responds by displaying:

(status)

DTc #

POrt #,#,#

(carriage return to exit) ?

The parameter for **POrt** is DTC, SIC and port number.

DTc displays the system's DTC's. It provides the same information provided when **TERMDSM** is initiated.

Help displays information about **TERMDSM**.

EXit terminates the **TERMDSM** subsystem.

Additional Discussion

For more information refer to *Troubleshooting Terminal, Printer, and serial Device Connections* (32022-61002).

untic

The **untic** utility decompiles the **terminfo** binary file.

Operation

The **untic** utility decompiles a **terminfo** binary file into its source format. If a **TERMINFO** environment variable is set, the **untic** utility searches the specified directory; otherwise, **untic** assumes the file is in the directory **/usr/lib/terminfo**. The output of an **untic** decompile is sent to the standard output.

Syntax

```
untic.hpbin.sys [term]
```

Parameters

term is the name of the terminal (default is the terminal from the **TERM** environment variable.)

Example

```
untic.hpbin.sys ansi
```

Related Information

Commands **tic** utility, **term**, **terminfo** database

V7FF8CNV

V7FF8CNV converts text and literals in VPLUS/XL forms files from a Hewlett-Packard 7-bit national substitution character set, to ROMAN8. V7FF8CNV is a special version of FORMSPEC.PUB.SYS and is run the same way.

Operation

1. Use the **STORE** command or **SYSGEN** to back up the forms file.
2. Configure your terminal for 8-bit operation. Refer to *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042) for information on peripheral configuration.
3. RUN V7FF8CNV.PUB.SYS, stepping through each form, field definition, save field, and function key label. As each screen is presented on the terminal, 7-bit substitution characters have already been converted to their ROMAN8 equivalent.
4. If the data is correct, press **Enter** and proceed to the next screen. If not, correct the data, then press **Enter** to continue.
5. After all screens are converted, recompile the forms file as usual.

Conversion applies to substitution characters found in all source record VPLUS/XL forms files with the following exception: substitution characters for "[" and "]" are not converted in screen source records since these indicate start and stop of data fields. The following would be converted:

- Text in screens.
- Function key labels.
- Initial values in save field definitions.
- Initial values in field definitions.
- Literals in processing specifications.

Alternate Character Sets

Hewlett-Packard block mode terminals, which have the capability of handling all or part of ROMAN8, can be divided into two groups. The group differentiation is based on how they handle alternate character sets when configured for 8-bit operation.

Files using alternate character sets on one group of terminals will not display correctly on the terminals of the other group, even when terminals from both groups are configured for 8-bit operation.

The use of characters from an alternate set affects the conversion procedure. If the forms file does contain characters from an alternate character set, choose one of the following alternatives:

- Eliminate the use of alternate character sets (either with **FORMSPEC** or while running **V7FF8CNV**).
- Define alternate character sets to appear correctly on Group 1 terminals. This happens automatically when **V7FF8CNV** is run from a Group 1 terminal. Characters from these alternate sets will appear as USASCII characters on a Group 2 terminal.

Group 1 - HP 2392A, 2625A, 2627A, 2628A, 2700, and 150

Use shift-out and shift-in characters to switch back and forth between an 8-bit base character set and an 8-bit alternate character set. This is standard for new Hewlett-Packard terminals and printers.

Group 2 - HP 2622A, 2623A, 2626A, and 2382A

(Do not use an HP 2624A or HP 2624B, as they are unable to handle 8-bit characters properly.) Group Two terminals use the eighth bit to switch back and forth between a 7-bit base character set and a 7-bit alternate character set. It is not possible to get true 8-bit operation (ROMAN8) and use an alternate character set (for example, Line Draw) at the same time because the base character set is not really 8-bit, but 7-bit with the additional characters defined in the alternate character set. Using both 8-bit ROMAN8 characters and Line Draw in the same file is not recommended since the user must continually redefine the alternate character set, switching back and forth between Roman Extension and the line drawing character set. Shift-out and shift-in are ignored by the terminal and return to the alternate character set when the high-order bit is on.

Procedure Example

V7FF8CNV must be run on a terminal supported by VPLUS/XL which supports display of all characters, enhancements, and alternate character sets used in the forms. If alternate character sets are used, the HP 2392, 2625, 2627, 2628, 2700, or 150 are recommended.

The V7FF8CNV procedure is:

1. Configure your terminal type properly for 8-bit operation by using the settings recommended in *System Startup, Configuration, and Shutdown Reference Manual* (32650-90042).
2. Run V7FF8CNV.PUB.SYS. Respond to prompts for the terminal group and the national substitution set.
3. Press **(Next)** to begin going through the forms file.
4. Press **(Enter)** after each screen until the end of the forms file is reached. Two exceptions are:
 - a. Enter Y in **Function key labels** on each FORM MENU and the GLOBALS MENU to see and convert function key labels.
 - b. On the field definition screen, if the processing specifications have converted data which you want to save, press the FIELD TOGGLE key, then **(Enter)** to save that conversion.

If you try to redisplay a screen which has already been converted and this conversion has been saved by pressing **(Enter)**, a message **Form contains 8 bit data** will be displayed. Do not press **(Enter)** again, but continue through the forms file.

5. Compile your forms file as usual.

VERSION

VERSION displays information about compatibility mode (CM) program files and native mode (NM) executable files (program files or executable library files), object files, and nonexecutable library files.

Operation

To invoke VERSION enter:

VERSION

or

VERSION *file(s)*

or

VERSION "*file(s),search string*"

Parameters

<i>file(s)</i>	The name of a program file or a wildcarded fileset.
<i>search string</i>	The name of a particular \$version string in a system object module SOM. (Not applicable for CM program files.) Quotes are required if a search string is specified. Spaces within the search string are significant and the search string is not case sensitive.

Example

To find out the version of the file P01P346A.PUB.SYS, enter:

VERSION P01P346A.PUB.SYS

Or, using a file set and a search string, you would enter:

VERSION "P@.@.MFGACCT,HEADER"

If a *file(s)* or "*file(s),search string*" is not entered the version prompt VERSION> appears and you may enter data then.

To terminate VERSION enter:

EXIT or :

If the input to VERSION is a file set, every file in the set will be processed even if an error occurs processing a previous file. If there is an error opening a file, the file system error will be displayed in addition to the VERSION error message.

VERSION displays the following information.

CM program file	Number of segments, stack size, maximum data segment size, total data reference base (DB), data segment limit (DL), and capabilities.
NM executable file	Unsatisfied procedure name, library search list, capabilities, maximum stack size, maximum heap size, entry name, and all \$version strings found in the SOMs.
NM object file	All \$version strings found in the SOMs.
NM nonexecutable library file	All \$version strings found in the SOMs.

Additional Discussion

For more information on the \$version strings found in the SOM, refer to *HP Pascal/iX Migration Guide* (31502-90004).

Index

- A**
 - account(s)
 - migrating with BULDACCT, 4-1
 - African peripherals, 16-1
 - application code written in both native and compatibility mode
 - facilitating with SWAT, **2-40**
 - Asian peripherals, 16-1
 - ASOCIATE.PUB.SYS, 2-2, 3-1
 - ASOCTBL, 2-2, 3-1-4
 - errors, **3-4**
 - input to, **3-2**
 - listing association table, **3-3**
 - parameters, **3-1**
 - removing entries from ASOCIATE.PUB.SYS, **3-3**
 - using wildcards with, **3-3**
 - ASSOCIATE command, 2-2, 3-1
 - association table, 3-1
- B**
 - backing up your system, 2-39
 - BACKOUT, 22-6
 - boot image, 20-5
 - SOMs, system libraries, **2-32**
 - breakpoint
 - setting within programs, 2-7
 - BUILDINT, 2-5
 - BUILDPME, 20-5
 - BULDACCT, 2-4ff, 4-1-6
 - migrating an account to a non-system volume, **4-3**
 - migrating groups to a non-system volume set, **4-4**
 - output from, **4-2**
 - processing options, **4-1**
 - recreating entire directory structure, **4-3**
 - recreating selected accounts, **4-3**
 - UDC files, **4-2**
- C**
 - CATALOG.PUB.SYS, 2-22
 - catalog(s)
 - modifying or generating with GENCAT, **2-15**
 - user defined, **2-22**
 - CHECKALL, 9-5
 - CHECKDIRC, 9-2, 9-5
 - CHECKDIREC, 9-5
 - CHECKEXTENTS, 9-3, 9-5
 - CHECKFILE, 9-4
 - CHECKLABEL, 9-3
 - CICAT.PUB.SYS, 2-22

- CLKUTIL, 2-6, 22-18
- code
 - modifying with PATCH, **2-30**
- compatibility mode code
 - managing with SEGMENTER, **2-33**
- compatibility mode program files
 - obtaining information about with VERSION, 2-48
- configuration
 - identifying with IOMAP, **2-17**
 - system, **2-41**
- CONFIGURE command in DISCUTIL, 6-3
- converting files to ROMAN8, 2-24
- copying files with FCOPY, 2-13, 8-2
- creating device link files, 14-1
- creating fifo files, 2-23, 14-1
- creating special files, 2-23, 14-1
- creating streams files, 14-1

D

- data
 - transport via SYSDUMP tape, **2-8**
- database load, 2-16
- data communications
 - configuring with NMMGR, **2-27**
- data stacks and registers
 - modifying, **2-7**
- DBLOAD.PUB.SYS, 10-1
- DEBUG, 2-7
- DEBUG, 9-5
- device class/user association table , 2-2, 3-1
- device link files, 14-1
- diagnostics in TERMDSM, 28-2
- directory
 - reproducing with BULDACCT, **4-3**
 - storing to tape, **2-39**
- directory check, 2-14, 9-1
- DIRMIG, 2-8
- DISASSOCIATE command, 2-2, 3-1
- DISCFREE, 2-9, 5-1-3
 - allocation summary, **5-2**
 - format options, **5-1**
 - histogram, **5-2**
 - parameters, **5-1**
- DISCUTIL, 2-10, 6-1-10
 - commands, **6-2**
 - CONFIGURE command, **6-3**
 - DISMOUNT command, **6-3**
 - DO command, **6-3**
 - DSTAT command, **6-4**
 - EXIT command, **6-4**
 - HELP command, **6-4**
 - LISTREDO command, **6-5**
 - messages and errors, **6-10**
 - MOUNT command, **6-5**
 - PDEV command, **6-6**

- REDO command, **6-6**
- SAVE command, **6-7**
- SHOWDEV command, **6-9**
- TAPE command, **6-9**
- UNCONFIG command, **6-9**
- DISCUTIL”
 - file recovery, **6-1**
- disk fragmentation, 2-9, 5-1
- disk management, without MPE/iX, 2-10
- disk space
 - evaluating free, **2-9**, 5-1
 - limits, **2-9**, 5-1
- disk volume(s), 4-1
- DISMOUNT command in DISCUTIL, 6-3
- DISPLAY, 22-8
- DISPLAYEXTENTS, 9-6
- DISPLAYLOCKFILE, 9-7
- DO command of DISCUTIL, 6-3
- DSTAT command, 2-9, 5-1
- DSTAT command in DISCUTIL, 6-4
- DTC, 28-2
- dump
 - in TERMDISM, **28-2**
 - taking, **2-11**
- DUMP, 2-11

E

- EBCDIC, 8-2
- EDIT/3000, 2-12, 7-1-4
 - commands, **7-2-4**
 - work file, **7-1**
- EMPTYSLUGH, 9-8
- executable library(s), 12-1
- EXIT, 9-8, 20-6, 22-9
- EXIT command in DISCUTIL, 6-4
- EXTENTDISTRIB, 9-9

F

- FCOPY
 - copying a subset of a file, **8-2**
- FCOPY, 2-13, 8-1-6
 - copying EBCDIC tape, **8-2**
 - copying IBM tape, **8-2**
 - defining devices with file equations, **8-5**
 - from* and *to* files, **8-2**
 - functions, **8-4**
 - general guidelines, **8-5**
 - hexadecimal, **8-2**
 - input files, **8-2**
 - output files, **8-2**
 - using formal file designators in, **8-5**
 - using with KSAM files, **8-6**
- fifo files, 2-23, 14-1
- file directories
 - checking, directory, label, extent, **2-14**
- file extent map display, 2-14, 9-1

- file recovery, DISCUTIL, 6-1
- files
 - merging, 23-1
 - restoring , 2-39
 - sorting, 23-1
 - storing , 2-39
- FIND, 22-9
- FINDSYM, 20-6
- formal file designators, 8-5
- FSCHECK, 2-14, 9-1-13
- fundamental operating system, 2-3

G GENCAT, 2-15

H

- hardware clock, 2-6
- hardware configuration
 - viewing with SYSMAP, **2-42**
- HELP, 9-10, 20-7, 22-10
- HELP command in DISCUTIL, 6-4

I

- I7DB8CNV, 2-16, 10-1-3
- IBM tape
 - copying with FCOPY, 8-2
- IMAGE data base
 - converting to ROMAN8, **2-16**, 10-1-3
- Initial stack, modifying, 18-4
- intrinsic disk files
 - building,changing, **2-5**
- IOMAP, 2-17
- ISL> prompt, 6-1

K

- KSAM, 15-1
 - copying files, **8-6**
- KSAM files
 - managing with KSAMUTIL, 2-18
- KSAMUTIL, 2-18

L

- labeled tape, 8-2
- label table check, 2-14, 9-1
- label table extent blocks, 6-7
- LANGINST, 2-19, 11-1-12, 16-2
 - add a language, 11-2
 - delete a language, 11-3
 - error messages, **11-9**
 - modifying 16-bit Asian ASCII/EBCDIC translation tables, **11-8**
 - modifying ASCII/EBCDIC translation tables, **11-6**
 - modifying local language formats, 11-3
- languages
 - displaying configured, 17-1
- language specific information
 - configuring onto your system, **2-19**
- libraries
 - creating and maintaining with LINK EDITOR/XL, **2-20**

- LINK, 22-3
- LINK EDITOR/XL, 2-20, 12-1-3
 - commands, **12-2-3**
- LIST, 22-10
- LISTFILE command, 13-1
- LISTREDO, 9-10
- LISTREDO command in DISCUTIL, 6-5
- LOADCME, 20-7
- LOADMILLI, 20-7
- LOADSXRT, 20-8
- LOADSYSLIB, 20-8
- LOG, 9-10, 13-4, 22-11
- log files, 13-1
 - managing with LOGTOOL, **2-21**
 - record types, **13-2**
 - system, memory, **13-1**
- LOGTOOL, 2-21, 13-1-4
 - command summary, **13-2-3**
 - HELP command, **13-1**
 - LISTLOG parameter, **13-2**
 - OUTFILE parameter, **13-2**
 - TYPE parameter, **13-2**

M

- MAKECAT, 2-22
- MAP, 20-9
- MEMLOGP, 13-1
- memory dump, 2-11
- memory log file, 2-21
- MERGE, 2-36, 23-1-3
- message catalogs
 - accessing with MAKECAT, **2-22**
- Message Catalogs Programmer's Guide , 2-15
- Middle East/African peripherals, 16-1
- migrating code
 - using OCT to convert compatibilty mode code to HP-PA, **2-29**
- migration
 - detecting problems in programs with OCA, **2-28**
 - from MPE V/E to MPE/iX, **2-8**
- Migration Process Guide , 2-8
- mirrored disk, 4-1
- MKNOD, 2-23, 14-1
- MODIFY, 22-13
- modify ASCII/EBCDIC Translation Tables, 11-6
- modifying 16-Bit Asian ASCII/EBCDIC Translation Tables, 11-8
- modifying files with EDIT/3000, 2-12
- MOUNT command in DISCUTIL, 6-5
- mounted volume table, 6-4
- MPE/iX Commands Reference Manual , 8-5

- N**
 - N7MF8CNV, 2-24, 15-1-2
 - eightth bit set, **15-1**
 - KSAM files, **15-1**, 15-2
 - user labels, **15-1**
 - native language I/O, 16-1
 - native language(s)
 - activating for devices, **2-25**
 - native language support, 17-1
 - verifying with NLUTIL, **2-26**
 - NLIODEF.PUB.SYS, 16-1
 - NLIOUTIL, 2-25, 16-1-4
 - commands, **16-1-4**
 - NLUTIL, 2-26
 - NLUTIL, 17-1
 - NMMGR, 2-27, 16-1
 - NOP instruction, 18-3

- O**
 - object code
 - analyzing with OCA, **2-28**
 - modifying and displaying, **18-1**
 - modifying with PATCH, **2-30**
 - OCA, 2-28
 - OCT, 2-29
 - OCTOMP, 2-29
 - OFFSET, 22-15
 - OPEN, 22-16

- P**
 - PASCAL/iX source code , 2-40
 - PATCH, 2-30, 18-1-6
 - D command, **18-2**
 - DG command, **18-4**
 - M command, **18-2**
 - MG command, **18-4**
 - subcommands, **18-2**
 - PATCHFILE, 22-16
 - PATCHJCW, 22-2
 - PDEV command in DISCUTIL, 6-6
 - permanent space, 5-1
 - PMAP, 21-2
 - preparing native mode object modules, 12-1
 - preparing Native Mode programs, 2-20
 - PREP command, 18-6, 21-2
 - PURGEFILE, 9-11
 - PXUTIL, 2-31, 19-1-3
 - PXUTIL utility, 2-31, 19-1

Q QUIT, 22-17

R REDO, 9-11
 REDO command in DISCUTIL, 6-6
 relocatable library(s), 2-33, 12-1
 reset in TERMDSM, 28-3
 RESTORE, 2-39
 ROMAN8, 2-16, 2-24, 2-47

S SAINT, 2-32, 20-1-13
 Commands, **20-9**
 Commands , **20-5ff**
 SAVE, 22-17
 SAVE command in DISCUTIL, 6-7
 security levels for TERMDSM, 28-1
 segmented library(s), 2-33
 modifying with SLPATCH, **2-34**
 SEGMENTER, 2-33
 semicolon (;), 22-6
 serial port connections
 modifying with TTUTIL, **2-45**
 SHOW, 22-18
 SHOWDEV command in DISCUTIL, 6-9
 SLPATCH, 2-34
 D command, **21-2**
 M command, **21-2**
 SLPLATCH, 21-1-3
 SL segment
 displaying, **21-2**
 modifying, **21-2**
 soft reset, 6-1
 SOM, 22-20
 modification, patching, **2-35**
 SOMPATCH, 2-35, 22-1-22
 SORT, 2-36, 23-1-3
 SORT-MERGE/XL, 23-1-3
 primary key, **23-2**
 secondary key, **23-2**
 terminating, **23-2**
 special files, 2-23, 14-1
 SPIFF, 2-37
 SPIFF, 2-37ff, 24-1-3
 commands, **24-1-2**
 SPL procedure head declarations, 2-5
 standalone utilities
 listing at ISL> prompt, 6-1
 standard, 25-1-13
 standards
 bootstrap, IPL, ISL, **2-38**
 STANDARDS, 2-38
 status in TERMDSM, 28-3
 \$STDLIST, 3-2
 STORE, 2-39
 storing and restoring files , 2-39

- streams files, 14-1
- SWAT, 2-40
- Switch Assist Tool, 2-40
- SWITCHLOG command, 13-1
- Switch Programming Guide , 2-40
- SXRT, 20-5
- SYMBOL, 22-20
- SYNCACCOUNTING, 9-12
- SYSDIAG, 13-1, 26-1
- SYSGEN, 2-41, 3-2, 16-1
- SYSMAP, 2-42, 26-1-2
 - CPUMAP, **26-1**
 - IOMAP, **26-1**
 - MEMMAP, **26-1**
- system backup and recovery , 2-39
- system diagnosis
 - TERMDSM, **2-43**
- system log file, 2-21
- system logging, 13-4
- system volume(s), 4-1

T

- TAPE command of DISCUTIL, 6-9
- TERMDSM, 2-43, 28-1-3
 - commands, **28-1-3**
 - security levels, **28-1**
- termtype file
 - modifying with TTUTIL, **2-45**
- tic, 2-44, 27-1
- tic utility, 2-44, 27-1
- timestamps, 2-6
- TOTALEXTENTS, 9-12
- transient space , 5-1
- translating files with FCOPY, 2-13
- TTUTIL, 2-45

U

- UDC files, 4-2
- UNCONFIG command of DISCUTIL, 6-9
- UNLOCKFILE, 9-13
- untic, 2-46, 29-1
- untic utility, 2-46, 29-1
- USE, 9-13, 22-2, 22-22
- USERLANG, 16-2
- user subprogram library(s), 2-33
- user volume(s), 4-1
- utilities, 2-4-46
 - PXUTIL, 2-31, 19-1
 - tic, 2-44, 27-1
 - untic, 2-46, 29-1

- V**
 - V7FF8CNV, 2-47, 30-1-3
 - alternate character sets, **30-2**
 - VERSION, 2-48, 22-22, 31-1-2
 - information displayed, **31-2**
 - volume label, 6-7
 - volume number in volume set, 6-6
 - volumes
 - mounted, **2-9**, 5-1
 - volume set(s)
 - managing with VOLUTIL, **2-49**
 - VOLUTIL, 2-49
 - RECOVER, **2-10**
 - VOLUTIL''
 - RECOVER, **6-1**
 - VPLUS
 - converting to ROMAN8, **2-47**

