

**DOMAIN System  
Command Reference**

**Order No. 002547  
Revision 04**

Apollo Computer Inc.  
330 Billerica Road  
Chelmsford, MA 01824

Copyright © 1987 Apollo Computer Inc.

All rights reserved.

Printed in U.S.A.

First Printing: May, 1983  
Latest Printing: December, 1986  
Updated: June, 1987

This document was produced using the SCRIBE document preparation system. (SCRIBE is a registered trademark of Unilogic, Ltd.)

APOLLO and Domain are registered trademarks of Apollo Computer Inc.

3DGR, Aegis, D3M, DGR, Domain/Access, Domain/Ada, Domain/Bridge, Domain/C, Domain/ComController, Domain/CommonLISP, Domain/CORE, Domain/Debug, Domain/DFL, Domain/Dialogue, Domain/DQC, Domain/IX, Domain/Laser-26, Domain/LISP, Domain/PAK, Domain/PCC, Domain/PCC-Remote, Domain/PCI, Domain/SNA, Domain/X.25, DPSS/MAIL, DSEE, FPX, GMR, GPR, GSR, Network Computing Kernel, NCK, Network Computing System, NCS, Open Network Toolkit, Open System Toolkit, OST, Personal Workstation, and Series 3000 are trademarks of Apollo Computer Inc.

Ada is a registered trademark of the U.S. Government (Ada Joint Program Office). PostScript is a trademark of Adobe Systems Incorporated.

Apollo Computer Inc. reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should in all cases consult Apollo Computer Inc. to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF APOLLO COMPUTER INC. HARDWARE PRODUCTS AND THE LICENSING OF APOLLO COMPUTER INC. SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN APOLLO COMPUTER INC. AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY APOLLO COMPUTER INC. FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY BY APOLLO COMPUTER INC. WHATSOEVER.

IN NO EVENT SHALL APOLLO COMPUTER INC. BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATING TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF APOLLO COMPUTER INC. HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

THE SOFTWARE PROGRAMS DESCRIBED IN THIS DOCUMENT ARE CONFIDENTIAL INFORMATION AND PROPRIETARY PRODUCTS OF APOLLO COMPUTER INC. OR ITS LICENSORS.

**Update Package 1 to the  
DOMAIN System  
Command Reference**

**Order No. 009496  
Revision 00**

Apollo Computer Inc.  
330 Billerica Road  
Chelmsford, MA 01824

Copyright © 1987 Apollo Computer Inc.

All rights reserved.

Printed in U.S.A.

First Printing: June, 1987

This document was produced using the SCRIBE document preparation system. (SCRIBE is a registered trademark of Unilogic, Ltd.)

APOLLO and Domain are registered trademarks of Apollo Computer Inc.

3DGR, Aegis, D3M, DGR, Domain/Access, Domain/Ada, Domain/Bridge, Domain/C, Domain/ComController, Domain/CommonLISP, Domain/CORE, Domain/Debug, Domain/DFL, Domain/Dialogue, Domain/DQC, Domain/IX, Domain/Laser-26, Domain/LISP, Domain/PAK, Domain/PCC, Domain/PCC-Remote, Domain/PCI, Domain/SNA, Domain/X.25, DPSS/MAIL, DSEE, FPX, GMR, GPR, GSR, Network Computing Kernel, NCK, Network Computing System, NCS, Open Network Toolkit, Open System Toolkit, OST, Personal Workstation, and Series 3000 are trademarks of Apollo Computer Inc.

Ada is a registered trademark of the U.S. Government (Ada Joint Program Office). PostScript is a trademark of Adobe Systems Incorporated.

Apollo Computer Inc. reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should in all cases consult Apollo Computer Inc. to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF APOLLO COMPUTER INC. HARDWARE PRODUCTS AND THE LICENSING OF APOLLO COMPUTER INC. SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN APOLLO COMPUTER INC. AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY APOLLO COMPUTER INC. FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY BY APOLLO COMPUTER INC. WHATSOEVER.

IN NO EVENT SHALL APOLLO COMPUTER INC. BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATING TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF APOLLO COMPUTER INC. HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

THE SOFTWARE PROGRAMS DESCRIBED IN THIS DOCUMENT ARE CONFIDENTIAL INFORMATION AND PROPRIETARY PRODUCTS OF APOLLO COMPUTER INC. OR ITS LICENSORS.

## Updating Instructions

The information in this package supersedes the contents of the *DOMAIN System Command Reference*, Revision 04. To update your manual, remove the old pages and insert the new pages as listed below. Insert this instruction sheet behind the title page as a record of the changes.

<u>Remove</u>	<u>Insert</u>
Title/Notice	Title/Notice
Preface	Preface
	2-14.1
	2-50.1
4-39/4-40	4-39/4-39.1 through 4-40
4-171/4-172 through 4-174	4-171/4-172 through 4-174
4-249/4-250	4-249/4-250
	4-263.1/4-263.2
4-277/4-278	4-277/4-278
4-315/4-316	4-315/4-316 through 4-316.1
4-319/4-320 through 4-330	4-319/4-320 through 4-330
Index-1 through Index-9	Index-1 through Index-8
Reader's Response/ Business Reply	Reader's Response/ Business Reply



## Preface

This document updates the SR9.5 version of the *DOMAIN System Command Reference* for software features included in the SR9.6 release. The *DOMAIN System Command Reference* is the third volume in the three-volume introduction to the DOMAIN computing system. The first volume, *Getting Started With Your DOMAIN System* (Order Number 002348), provides a tutorial approach to getting started on your node. The second volume, *DOMAIN System User's Guide* (Order Number 005488), constitutes a handbook that takes you beyond the introductory stage into practical applications of Display Manager (DM) and Shell operations. This third document provides complete reference information on all of the DM and Shell commands that are available to you. We assume that you are familiar with the material in the first two books before you attempt to use this reference manual. Fundamental concepts like file structure and usage are taken for granted here. We tell you *how* to use the commands; not *why* you might want to use them.

For information on creating, protecting, and maintaining the network environment, see *Administering Your DOMAIN System* (Order number 001746). For information on creating, protecting, and maintaining the DOMAIN/BRIDGE internet environment, see *Administering Your DOMAIN/BRIDGE Internet* (Order number 005694).

### Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. In order to make it easy for you to communicate with us, we provide the User Change Request (UCR) system for software-related comments, and the Reader's Response form for documentation comments. By using these formal channels you make it easy for us to respond to your comments.

You can get more information about how to submit a UCR by consulting the description of the Shell command CRUCR (CREATE\_USER\_CHANGE\_REQUEST). You can also get more information by typing:

```
$ HELP CRUCR <RETURN>
```

For your comments on documentation, a Reader's Response form is located at the back of this Guide.

## Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

- UPPERCASE** Uppercase words or characters in formats and command descriptions represent commands or keywords that you must use literally.
- lowercase** Lowercase words or characters in formats and command descriptions represent values that you must supply.
- [ ]** Square brackets enclose optional items in formats and command descriptions. In sample Pascal statements, square brackets assume their Pascal meanings.
- { }** Braces enclose a list from which you must choose an item in formats and command descriptions. In sample Pascal statements, braces assume their Pascal meanings.
- |** A vertical bar separates items in a list of choices.
- < >** Angle brackets enclose the name of a key on the keyboard.
- CTRL/Z** The notation CTRL/ followed by the name of a key indicates a control character sequence. You should hold down the <CTRL> key while typing the character.
- ...** Horizontal ellipsis points indicate that the preceding item may be repeated one or more times.
- Vertical ellipsis points mean that irrelevant parts of a figure or example have been omitted.

## Summary of Technical Changes

### The Display Manager

The following new features are available in the SR9.6 DM.

#### *New Commands and Features*

- CDM (CHANGE\_DISPLAY\_MANAGER\_MODE): Change the display manager mode.
- LCM (LOAD\_COLOR\_MAP): Load a color map.

### The Shell

The following new features are available in the SR9.6 shell.

#### *New Commands and Features*

- SCRCH (SHOW\_SCREEN\_CHARACTERISTICS): Show screen characteristics.
- SH8 (INVOKE\_8-BIT\_SHELL): Invoke 8-bit shell.

#### *Changes to Existing Commands*

- CPSCR: New -APPEND and -GPR[\_BITMAP] options.
- LCNET: New -C option.
- RTSTAT: New -DESC[RIPTION] option.
- VCTL: New -SUSP, -EOL, -[NO]RAW, -[NO]ENABLE\_SIGS options.
- VSIZE: New -STD option.
- VT100: Support for PASTE and F8 keys.
- WBAK: New -PDTU option.

NOTE: A vertical bar in the margin indicates a substantive technical change from Revision 04 of the *DOMAIN System Command Reference*.



## Summary of Technical Changes from SR9.2

### Display Manager

The following new features are available in the SR9.5 Display Manager.

#### New Commands

- BL (BALANCE): Balance delimiters.
- CURS (CURSOR\_LOCK): Control cursor positioning.
- MONO (MONOCHROME): Set color monitor to black and white.

#### Changes to Existing Commands

- CP: -N no longer must be the last token on the command line.
- CPO: New option, -W, to create process in wait mode.
- CPS: New option, -W, to create process in wait mode.
- DQ: New argument, WINDOW\_NAME, which allows you to send a fault to a specific process in a named window.
- WC: New argument, WINDOW\_NAME, which allows you to close a specific named window.

### Shell

The following new features are available in the SR9.5 Shell.

#### New Commands

- SHUTSPM: Shut down the Server Process Manager on a node.
- UMASK: Set DOMAIN/IX file-creation mode mask.

#### Changes to Existing Commands

- BIND: -SET\_VERSION nn.mm option is no longer valid.
- CPF: New option, -NSUBS; -SUBS is the default.
- CPT: New option, -NSUBS; -SUBS is the default.
- CRP: New option, -QUIET, to suppress connection messages.
- CRTYOBJ: New option, -PRE\_SR9.5, to generate object modules in pre-SR9.5 format.
- DEBUG: New options, -NSRC to suppress source file display, and -GLOB to enable DEBUG to enter routines in global address space.
- EDACL: New options, -SETPERS, -SETPROJ, -SETORG, -DYNAMIC, for DOMAIN/IX compatibility.
- EDFONT: font pathname and initial editing character may now be passed to the program as command line arguments.
- EDNS: New interactive command, UPDATE, to update the NS\_HELPER database on a remote node.
- FPPMASK: Several new floating-point error mask conditions may now be reported.
- HPC: Several new options: -TO name, -FROM name, -PROC name, -LIMIT n, -MAP, -NHDR.
- ITEST: Command renamed to IOS\_TEST.

- LD: Several new options: -SC, -SR, -W, -CRB, -CRA, -USB, -USA, -MOB, -MOA. LD also now automatically sizes its output to the width of the transcript pad's window.
- LUSR: New option, -IDLE, to include idle nodes in report listing.
- OLD\_EDFONT: Obsolete command; no longer supported.
- PRF: Several new options: -COL, -POST, -LPI, -PAPER\_SIZE, -ORIENT, -C, -NC, -D, -ND.
- RTCHK: -DEVICE option replaces -IIC.
- SET: Default startup file pathname is now USER\_DATA/SH/STARTUP in the login home directory.
- SH: Default startup file pathname is now USER\_DATA/SH/STARTUP in the login home directory.
- SIORF: -OBJ is obsolete and no longer supported. The command now handles binary objects automatically.
- SIOTF: -OBJ is obsolete and no longer supported. The command now handles binary objects automatically.
- TB: Complete overhaul; now performs cross-process traces.

NOTE: A vertical bar in the margin indicates a substantive technical change from Revision 03.

## Summary of Technical Changes

### The Shell

The following new features are available in the SR9.2 Shell.

#### *New Commands and Features*

- CRTY (CREATE\_TYPE): Create a new type.
- CRTYOBJ (CREATE\_TYPE\_OBJECT): Create a type object module for binding.
- DLTY (DELETE\_TYPE): Delete a type.
- EDIP (EDIT\_IP\_HELPER): Invoke editor for IP\_HELPER.
- INTM (INSTALL\_TYPE\_MANAGER): Install a type manager.
- INTY (INSTALL\_TYPE): Install a new type.
- ITEST (IOS\_TEST): Test IOS\_\$calls.
- SETVAR (SET\_VARIABLE): Set the value of a variable.

#### *Changes to Existing Commands*

- KBD: New keyboard type added.
- LTY: Change in command syntax for -N option.

NOTE: A vertical bar in the margin indicates a substantive technical change from Revision 03 of the *DOMAIN System Command Reference*.

C

C

C

C

C

## Summary of Technical Changes

### Display Manager

The following new features are available in the SR9.0 Display Manager.

#### New Commands

- ENV (ENVIRONMENT): Set or display an environment variable.
- TNI (TO\_NEXT\_ICON): Move cursor to next icon.

#### Changes to Existing Commands

- LO [-ON|-OFF]: Enable/disable logoff capability.
- SQ (SEARCH\_QUIT): Now identical to ABRT.
- WP (WINDOW\_POP): Now works with named windows and window groups.

### Shell

The following new features are available in the SR9.0 Shell.

#### New Commands and Features

- Active functions: You may now use the construct `^"command"` to assign the value returned by `'command'` to a Shell variable. See the *DOMAIN System User's Guide* for details.
- DSPST (DISPLAY\_PROCESS\_STATUS): Display process status graphically.
- EDNS (EDIT\_NAMING\_SERVER): Invoke naming server editor.
- EXPORT: Change a Shell variable into an Environment variable.
- FOR: Execute a FOR statement.
- HLPVER (HELP\_VERSION): Provide HELP support for Shell scripts.
- LTY (LIST\_TYPES): List installed types.
- NOT: Negate a Boolean value.
- OLD\_EDFONT: Former version of EDFONT with function key interface.
- PRF (PRINT\_FILE): Now supports Imagen 8/300 and Imagen LBP-10 laser printers connected via SIO lines or the Multibus.
- PRFD (PRF\_DISPLAY): Invoke menu-based PRF.
- SELECT: Execute a SELECT statement.
- SEND\_ALARM: Send messages to alarm servers.
- SET: Set current Shell conditions.
- SOURCE: Execute a Shell script at the current Shell level.
- VSIZE (VT100\_SIZE): Set or display VT100 window settings.

#### Changes to Existing Commands

- ARGS -ERRROUT: Now writes output to error output if desired.
- BIND: Several new options: -ALLRESOLVED, -BDIR, -LOCALSEARCH, -NOLOCALSEARCH, -MULTIRES, -NOMULTIRES, -MAKERS, -NUND, -SET\_VERSION, -SYSTYPE.
- CPBOOT: Now supports cartridge tapes.

- CPF -DU: Now deletes currently locked files once they are unlocked.
- CPT -F: Force tree deletion if you have P rights.
- CRP: New option: -ME; new functionality in -LOGIN; -MBX no longer available.
- CRUCR: New corporate address for UCR submissions.
- CTNODE: Several new options: -FROM, -ON, -MS, -MD, -IDUPL, -LC, -ROOT, plus three new paragraphs in the description explaining "confusion resolution".
- DEBUG: New information on macros and variables, new option, -PROC, for cross-process debugging
- EDACL: New option, -UNIX, plus new directory rights "S" (search) and "E" (expunge).
- EDFONT: New version with a pointing device-driven interface.
- EDMTDESC: Now supports cartridge tapes.
- EMHASP: No longer standard software. Documentation is now provided with the RJE product.
- EXISTF: Wildcarding in pathname arguments is now supported.
- EXIT: Now works for FOR and SELECT as well as WHILE.
- FPAT: New options: -RM, -RMF, -LM.
- HASPSVR: No longer standard software. Documentation is now provided with the RJE product.
- INLIB: Now supports multiple pathnames and wildcarding.
- INVOL: Default paging size (option 8) is now 352 pages instead of 256. A new option 10 allows you to set or display the sector interleave factor for a volume. (See Appendix D.)
- LAS: New options: -ALL, -FROM, -TO, -PROCESS.
- LCNODE: New options: -C, -ID.
- LD -ROOT: Supports new naming server.
- NETSVC -S: Controls network server availability on a node.
- NEXT: Now works for FOR and SELECT as well as WHILE.
- PRF: Completely overhauled for SR9.
- PROBENET: New options: -D, -SENS. Deleted option: -Q. Comment character changed from '(blank)' to '#'. Reported statistics also changed.
- PRSVR -N: Assign a name to the print server process.
- RBAK: Now supports cartridge tapes with the options -DEV CT, -RETEN, -NRETEN, -REWIND. Limited floppy disk support.
- READ: New option, -ERRIN, for reading from error input; new -TYPE choices: ANY and ENV.
- READC: New option, -ERRIN, for reading from error input.
- READLN: New option, -ERRIN, for reading from error input.
- RETURN -OK: Same as -TRUE.
- RWMT: Now supports cartridge tapes with the options -DEV CT, -RETEN, -NRETEN. Limited floppy disk support.
- SALVOL: Now allows controller unit number specification for use when two or more of the same storage device are attached to one node.
- SH: Completely overhauled for SR9.
- TCTL: Several new options: -[NO]RTS\_ENABLE, -FORCE, -NO[RAW].
- WBAK: Now supports cartridge tapes with the options -DEV CT, -RETEN, -NRETEN, -SYSBOOT, -NO\_EOT. Limited floppy disk support.

NOTE: A vertical bar in the margin indicates a substantive technical change from Revision 02.

# Contents

<b>Chapter 1 Display Manager Basics</b>	<b>1-1</b>
1.1. Defining Points and Regions	1-1
1.2. Defining Window Boundaries	1-3
1.3. Defining a Range of Text	1-4
1.4. Using Regular Expressions	1-4
1.5. Key Naming Conventions	1-8
1.5.1. Standard Key Names	1-8
1.5.2. Controlling Keys from Within a Program	1-11
1.6. Special Characters in DM Scripts and Key Definitions	1-11
<b>Chapter 2 Display Manager Commands</b>	<b>2-1</b>
<b>Chapter 3 Shell Basics</b>	<b>3-1</b>
3.1. Command Format	3-1
3.1.1. Arguments	3-1
3.1.2. Separators	3-2
3.1.3. Node Specifications	3-2
3.2. Using Special Characters	3-3
3.3. The Command Line Parser	3-4
3.3.1. Standard Command Options	3-6
3.3.2. Command Line Parser Options	3-6
<b>Chapter 4 Shell Commands</b>	<b>4-1</b>
<b>Index</b>	<b>Index-1</b>

## Illustrations

<b>Figure 1-1.</b>	The Low-Profile Model II Keyboard Map	1-10
<b>Figure 1-2.</b>	The Low-Profile Model I Keyboard Map	1-10
<b>Figure 1-3.</b>	The Mouse Key Map	1-10
<b>Figure 1-4.</b>	The 880 Keyboard Map	1-11
<b>Figure 3-1.</b>	Typical Shell Command Format	3-1

## Tables

<b>Table 3-1.</b>	Command Shell Special Characters	3-4
-------------------	----------------------------------	-----

# Chapter 1

## Display Manager Basics

This chapter summarizes the basic concepts that apply to the DM commands described individually in the following chapter. For a more indepth discussion of these concepts, please refer to the *DOMAIN System User's Guide*.

### 1.1. Defining Points and Regions

Unless otherwise noted, all DM commands must be preceded by a pointing operation. This generally involves moving the cursor to the spot where the command is to be executed (for example, pointing to the window that you want to scroll), or specifying a specific screen or line location as a command argument. If you don't specify some pointing function, the Display Manager executes the command at the current cursor position.

To point, simply move the cursor to the desired location. For example, to point to a window, place the cursor anywhere inside the window. The command reads the cursor position to determine which window you mean. (Please note that when you use the block cursor to designate a point on the screen, the designated point is actually at the lower left corner of the block cursor.)

You can also define a point in any of the following ways:

**line-number** Line numbers begin at 1 and range upward to the last line *in the pad*. Pads may contain up to 262,143 lines. You may use the symbol "\$" to refer to the last line in the pad. Remember that the edit pad window legend contains the line number of the top line in the window for reference. You may also display the line number (plus the column number and X and Y coordinates) of the current cursor position by using the DM command "=".

**+/- line-number** The "+/- line-number" format denotes the nth line before or after the current cursor position *in a pad*.

**[[line-number],[column-number]]** This format indicates the point by line and column number *in the pad*. The Display Manager assumes the current line if the first portion is omitted, and column one if the second portion is omitted. Line numbers range from 1 to the last line in the pad (262,143 max.). Column numbers range from 1 to 256. *When you specify a point in this format, you must use the outer set of square brackets to enclose the numbers.* This is how the Display Manager distinguishes between line/column positions in a pad and X/Y coordinates on the screen (below). Note that the use of "\$" to denote the last line in the pad does *not* work within square brackets.

Examples: [127,14] Line 127, column 14  
 [53] Line 53, column 1. Brackets are optional in this case: see above.  
 [.12] Column 12 of the current line

([x-coordinate],[y-coordinate])

Screen coordinates specify bit positions *on the display*. The origin (0,0) is at the extreme upper left corner of the screen. Maximum values for x and y coordinates depend on the size of the screen in use. The current x or y coordinate of the cursor is used if you omit it from the coordinate pair. When you specify a point in this format, you must use the outer set of parentheses to enclose the numbers. This is how the Display Manager distinguishes between line/column positions in a pad (above) and X/Y coordinates on the screen.

Examples: (200,450) Bit position with the given coordinates  
 (135) Bit position whose X coordinate is 135 and whose Y coordinate is the same as the current cursor position.  
 (.730) Bit position whose X coordinate is the same as the current cursor position and whose Y coordinate is 730.

/regular-expression/ or \regular-expression\

A regular expression specifies a string in the pad that begins or ends the region of interest. Regular expressions are described in section 1.4.

Now that we can identify points, let's turn to regions. A **region** is simply the area between two points. To define a region, you need to use the DM command DR (DEFINE\_REGION). The region definition operation has the following format:

[point] DR: [point]

The first point marks one corner of the region. The second point marks the opposite corner of the region. Remember that the points may be defined by cursor positions or specified explicitly in one of the alternate formats mentioned above.

For convenience, the predefined key, <MARK>, invokes the DR command. Point the cursor at the start of the range, MARK it, then point at the end of the range. That's all there is to it.

For those DM commands that require you to specify a region in which to operate, you can declare it either by MARKing it, or by explicitly specifying the region with one of the techniques described above. If a DM command does require you to define a region, the command is specified in the following format:

[region]command

The symbol [region] indicates where you must define the region. (The definition of a range for text editing operations -- cut, paste, substitute, etc.-- is slightly different. Refer to section 1.3 for more information.)

## 1.2. Defining Window Boundaries

When a window's size or position on the screen is changed in any way, the DM determines the new boundaries of the window using calculations based on a pair of points (a "point pair") on the screen. Usually, the first point in the pair has been defined with the DR command and the second point is the current cursor position, although you may provide absolute point coordinates as described in section 1.1.

Each point may specify either a new or existing edge of a window, or a new or existing corner of a window. The new window, then, is created based on the relationship between the X and Y coordinates of the two points. When either of the points specifies a new upper edge or right edge for a window, the position is adjusted to account for the size of the displayed block cursor, because the actual coordinates of the cursor are determined by its lower left corner. This adjustment is made only when the coordinate source is the block cursor, and not when the point comes from the touchpad or mouse, or from explicitly entered coordinates.

The relationship between the two points in the point pair affects the actions of the window-related commands CP, CE, CV, CC, WG, WM, and WDF in the following ways:

1. Horizontal movement only (y coordinates of the two points are equal):

Creation - Create a window bounded by the given x coordinates, the top of the screen, and just above the normal DM command window (i.e., a full vertical window).

WG/WM - Select the unobscured vertical edge nearest to the first point, and change the x coordinate of that edge to be that of the second point. The y coordinate of the first point must be within the unobscured range of y coordinates of the selected edge.

2. Vertical movement only (x coordinates of the two points are equal):

Analogous to horizontal movement, except that for creation, the full horizontal width of the screen is used.

3. No movement (two points are equal):

Creation - create a 512 by 512 window centered as nearly as possible (subject to screen boundaries) on the given cursor position.

WG - treated as in 4 below.

WM - Select the unobscured corner nearest the given point, and move the corner to that point.

4. Two points differ in both x and y:

Creation - The given four coordinate values form opposing corners of the window.

WG/WM - The first point selects the nearest unobscured corner (the corner itself must be visible) and that corner is repositioned at the second point.

If only one point is given (i.e., the DR command is not issued), grow is illegal and move behaves as in step 3 above. The DM uses one of its five default window regions, or a default determined by the last window creation or deletion (WC) command, as follows:

- If the last such command was window deletion (i.e., WC), the default region is the same as that of the deleted window.
- If the last such command was a successful window creation command, the default region is the next third of the screen following the created window.
- If the last such command was an unsuccessful window creation command, the default region is the same as specified in that command.

To define the five default window regions, use the DM command WDF.

### 1.3. Defining a Range of Text

The text editing commands that perform cut, paste, and substitute functions operate on a range of text. That range is declared just as you would mark any other region in a pad; i.e., you place the cursor at the start of the range, press <MARK>, then move the cursor to the end of the range and issue the command in question.

The region of text you define for a cut, paste, or search operation is highlighted in reverse video when you use the <MARK> key. This is because that key invokes the DR;ECHO command sequence. You can still use the DR command alone to place a mark, but the highlighting feature is not invoked without ECHO. You can cancel the defined range with the ABRT command (invoked with CTRL/X). Refer to the descriptions of the DR, ECHO, and ABRT commands for more information.

Please note that the character under the cursor at the end of the range is NOT included within the range. Note also that you may NOT declare a range explicitly as an argument to the editing commands, since those commands do not, in general, accept arguments. You must use the MARK key or the DR command sequence.

The default range is different for these editing operations, too. While the general DM default range is the current cursor position, cut, paste, and substitute commands apply to all characters from the current cursor position up to the end of the line (including the NEWLINE character) if no other range has been marked immediately prior to invoking the command.

### 1.4. Using Regular Expressions

Special "regular expression" notation is used to specify patterns for search and substitute strings in the Display Manager editor. This notation is also used in the Shell commands ED (EDIT), EDSTR (EDIT\_STREAM), FPAT (FIND\_PATTERN), FPATB (FIND\_PATTERN\_BLOCK), and CHPAT (CHANGE\_PATTERN). Regular expressions permit you to concisely describe textual patterns without necessarily knowing their exact contents or format. You can create expressions to describe patterns in particular positions on a line, patterns that always contain certain characters and sometimes include others, or patterns that match text of indefinite length.

Regular expressions are constructed as follows:

1. Any standard ASCII character (except those discussed below) is a regular expression and matches one and only one occurrence of that character. (For multiple occurrence

matches, see "\*" below.) The case of the characters in the expression is not significant by default. Use the DM command SC (SET\_CASE) to control case significance.

```
SAM
fred12      All valid expressions.
Joe(a&b)
```

2. A percent sign (%) at the beginning of a regular expression matches the empty string at the beginning of a line. If a "%" is not the first character in the expression, then it simply matches the percent character. Use this special feature to mark the start of a line in a regular expression.

"%Print" matches the string in line a but not line b, because in line b *Print* is not at the beginning of the line.

- (a) *Print* this file.
- (b) This *Print* file.

3. A dollar sign (\$) at the end of a regular expression matches the null character at the end of a line. If "\$" is not the last character in the expression, then it simply matches the dollar sign character. Use this special feature to mark the end of a line in a regular expression.

"file\$" matches the string in line a but not line b, because in line b *file* is not followed by an end-of-line marker.

- (a) Print this *file*
- (b) This *file* is permanent

4. A question mark (?) matches any *single* character except a NEWLINE character. The only exception to this is if the "?" appears inside a character class (see below), in which case it represents the question mark character itself.

"?OLD???" matches a and b, but not c, because in line c the letters "OLD" are alone on the line.

- (a) HOLDING
- (b) FOLDERS
- (c) OLD

5. Whereas the above expression (?) matches only a single occurrence of a pattern, an asterisk (\*) following a regular expression causes it to match zero or more occurrences of that expression. The only exception to this is if the "\*" appears inside a character class (see below), in which case it represents the asterisk character itself. Matching zero or more occurrences of some pattern is called a closure. An expression used in a closure will never match NEWLINE.

a\*b                    Match b, ab, aab, etc.

%a\*b                    Match any string that begins with a and ends with b, and that is also the first string in the line. Any number of other characters can come between a and b.

[A-Z][A-Z][A-Z]\* Match any uppercase word; i.e., any string containing at least two (and possibly more) uppercase characters (see character classes, below). Strings like "Mary" would *not* match, since "Mary" does not begin with *two* uppercase characters.

6. A string of characters enclosed in square brackets "[string]" is called a character class. This pattern matches any *one* character in the string but no others. If, however, the *first character* of the string is a tilde (~), the regular expression matches any one character *except* the characters in the string. If ~ is not the first character in the string, then it simply matches the tilde character. Note also that the other special characters % \$ ? \* lose their special meaning inside square brackets, and simply represent themselves.

[sam] Match the single characters s, a, or m. (If you want to match the word "sam", don't use the square brackets.)

[~sam] Match any single character except s, a, or m.

7. Within a character class, you can specify any one of a range of letters or digits by indicating the beginning and ending characters in the range separated by a hyphen. That is, 0 through 9 matches any single digit; a through z or A through Z matches any single letter, lowercase or uppercase respectively. (Remember, though, that the actual matching search ignores case unless you have used the DM command SC to enable case sensitivity.) The range can be a subset of the digits or letters (i.e., a through n or 3 through 8). The first and last characters of the range, however, must be of the same type: digit, lowercase letter, or uppercase letter. "[A-9]" is illegal.

Note that the "-" character has a special meaning inside square brackets. If you want to include the literal hyphen character in the class for matching, it must either be the first or last character in the class (so that it does not appear to separate two range-marking characters) or be escaped (see below).

The "]" character is also special to character classes -- it closes the class descriptor list. If you want to include the right bracket character in the class, it also must be escaped (see below).

In summary, the following characters have special meaning inside square brackets: ~  
- ]

[a-d] Match any single occurrence of a, b, c, or d.

%[A-Z] Match any capital letter that is also the first character on the line (%).

1-[1-9][0-9]\* Match any of the page numbers in this chapter.

[0A-Z] Match any string containing a zero *or* a capital letter.

[~a-z0-9] Match any uppercase letter or punctuation mark (i.e., no lowercase letter or number).

8. The at sign (@) is an escape character. Characters preceded by the "@" character have special meaning in regular expressions, as indicated below.

**@n** Match NEWLINE character.

**@t** Match a tab character. Note, however, that the keyboard TAB key does not insert a literal tab; rather it moves the cursor to the display's next tab position. In a regular expression, **@t** matches only tab characters that have been inserted with **@t**.

**@f** Match a form feed character.

In addition, the escape character may be used inside a character class definition (**[]**) to specify literal occurrences of characters like **"-"** and **"]"**, which have special functions inside square brackets. You may also use it whenever you need a literal occurrence of some special character in a normal expression (like **?**, **\***, or **@** itself).

**[A-Z@-@]** Match any capital letter, a hyphen, or a right bracket.

**@?@\*** Match a question mark followed by an asterisk, rather than zero or more occurrences of any character (**?\***).

9. You can concatenate regular expressions to form a more complex regular expression. The resulting regular expression matches the concatenation of the strings that the component regular expressions match. All of the examples above concatenate expressions (single characters of some sort) into longer strings for matching.

10. You can "tag" parts of a regular expression to help rearrange pieces of a matched string. A text pattern surrounded by braces **"{pattern}"** is remembered and can be referred to by **"@n"**, where **n** is a single digit referring to the string remembered by the **n**th pair of braces.

**s/{???}{?\*/@2@1/**

**S** is the DM command for string substitution. The example will move a three-character sequence from the beginning of a line to the end of the line. **"???"** matches the first three characters of the line, and **"?\*"** matches the rest of the line.

**so/{?}{?}/@2@1/** **SO** is also a DM command for string substitution, but it only substitutes the first occurrence of the first pattern on a line. The example will transpose two characters beginning with the one under the cursor. This can be a handy key definition if you often type **"ei"** for **"ie"**, etc.

## Summary of Features

c	Literal character
%	Beginning of line (if first character only)
\$	End of line (if last character only)
?	Any single character except NEWLINE
*	Closure (zero or more occurrences of previous pattern)
[...]	Character class (any one of these characters)
[^...]	Negated character class (all characters except those in brackets or NEWLINE)
[c1-c2]	Any one of a range of characters from c1 through c2 (must be same type)
@c	Escaped character (e.g., @%, @[, @*, etc.)
{expr}	Tagged expression for use later in command line

**CAUTION:** Remember that the special characters described above apply only to regular expression operations. Some of these characters also have meanings (often radically different) in Shell commands and other software products. If you are using a regular expression as a part of one of those Shell commands or products, be sure to enclose the expression in quotation marks so that it will not be misinterpreted.

## 1.5. Key Naming Conventions

Every key on your keyboard (and mouse) has a name; in fact, almost every key has a set of three or four names. One set is the normal one, and is invoked when the key is pressed down. The second set is invoked when the key is released; these names are the up-transition names. The third set is invoked when the key is pressed simultaneously with the SHIFT key; these are the shifted names. Finally, many keys have special functions when they are pressed simultaneously with the CTRL key; these are the control shifted names.

### 1.5.1. Standard Key Names

The definable keys (see Figures 1-1, 1-2, and 1-4) have the following names:

**Letters and numbers**                      These are named by their own single character. The capital letters are distinct from the lowercase letters: merely refer to A instead of worrying about "a shifted". When you refer to these keys in a key definition, enter them within single quotation marks.

**ASCII Control**                              These are the standard intraline and interline control keys.

```
CR : Carriage Return
BS : Back Space
TAB : Tab
TABS : Shifted Tab
^TAB : Control Shifted Tab
ESC : Escape (low-profile only).
      Same as '^[' (hex 1B).
DEL : Delete (low-profile only).
      Same as '^|' (hex 7F).
```

## Alphabetic Control

These are named  $\hat{x}$ , where  $x$  is some other valid key name (i.e.,  $\hat{Y}$  for CTRL/Y and  $\hat{N}$  for CTRL/N). There are also six non-alphabetic control characters. Their names must appear in single quotation marks. The names and the hexadecimal values of the keys are:  $\hat{[}$  (hex 1B),  $\hat{\backslash}$  (hex 1C),  $\hat{]}$  (hex 1D),  $\hat{\sim}$  (hex 1E),  $\hat{?}$  (hex 1F), and  $\hat{|}$  (hex 7F).

## DM Function

These keys perform special Display Manager functions. Those on the left side of the keyboard are named L1 through L9 and LA through LF. (Note that the low-profile keyboards have an extra row of keys below L1 through L3. These keys are named L1A, L2A, and L3A.) Their up-transition names are L1U through L9U and LAU through LFU. Their shifted names are L1S through L9S and LAS through LFS. The DM Function keys on the right side of the keyboard are named R1 through R6. Their up-transition and shifted names are formed in the same way that the left-side keys are.

**NOTE:** Due to internal hardware restrictions, the 880 keyboard does not execute shifted definitions for these keys. The DM will not object if you define shifted operations for them, but nothing will happen when you try to use the shifted definitions.

## Program Function

These keys are specially reserved for user program control. They appear at the top of the keyboard and are named F1 through F8, as labeled. Their up-transition names are F1U through F8U. Their shifted names are F1S through F8S. Their control shifted names are  $\hat{F}1$  through  $\hat{F}8$ . (Note that the low-profile Model II keyboard has two additional program function keys, F0 and F9. Their shifted and control shifted names are derived as described above.)

## Numeric Keypad

These keys are only available on the low-profile Model II keyboard. The keypad's numeric keys are named NP0 through NP9. The keypad symbols are named NP+, NP-, and NP., respectively. The "Enter" key is named NPE. Keys 0 through 9, plus (+), and minus (-) can have shifted names (for example, NP+S).

## Mouse

These are the keys located on the optional mouse pointing device (Figure 1-3). Their names are M1, M2, and M3. Their up-transition names are M1U, M2U, and M3U. There are no shifted or control shifted names.

Names containing special characters and all ordinary graphic characters must be in single quotes. For example, to define the lowercase  $x$  key so that it acts just like the uppercase  $X$  key, you would use the following command line:

```
kd 'x' es 'X' ke
```

Consequently, however, you may find it difficult to redefine the  $x$  key, because the Display Manager enters an  $X$  when you press the  $x$  key. This is true for all the alphanumeric and special

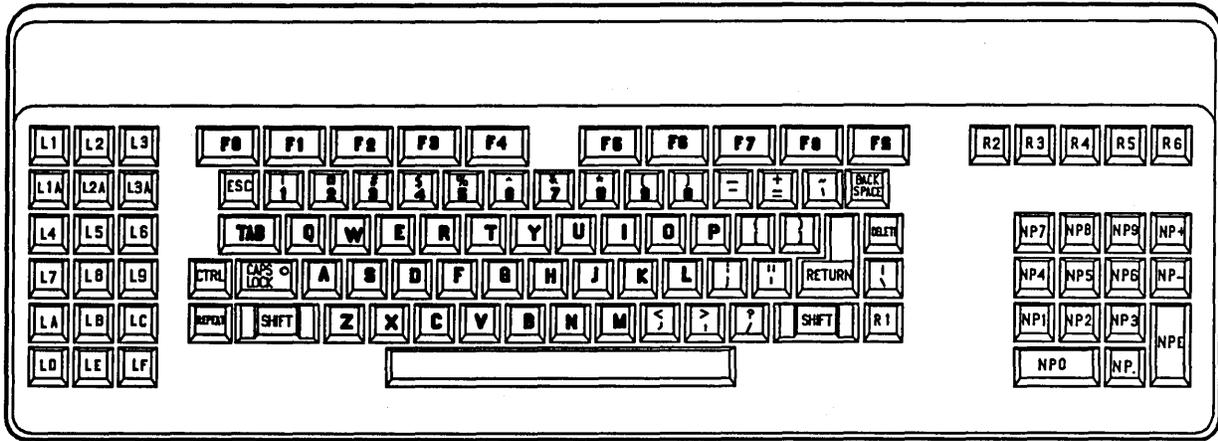


Figure 1-1. The Low-Profile Model II Keyboard Map

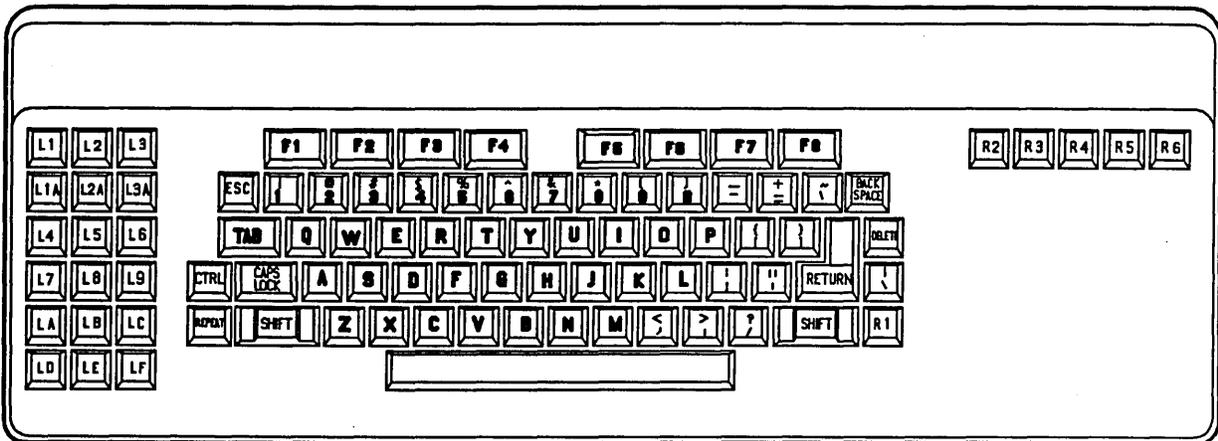


Figure 1-2. The Low-Profile Model I Keyboard Map

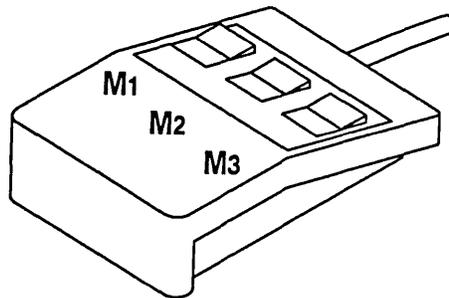


Figure 1-3. The Mouse Key Map

character keys. Although it is possible to change the definitions of these keys, that capability is intended mainly for use in programs. When a program defines a key, the definition applies only while the program is running and only in pads the program controls.

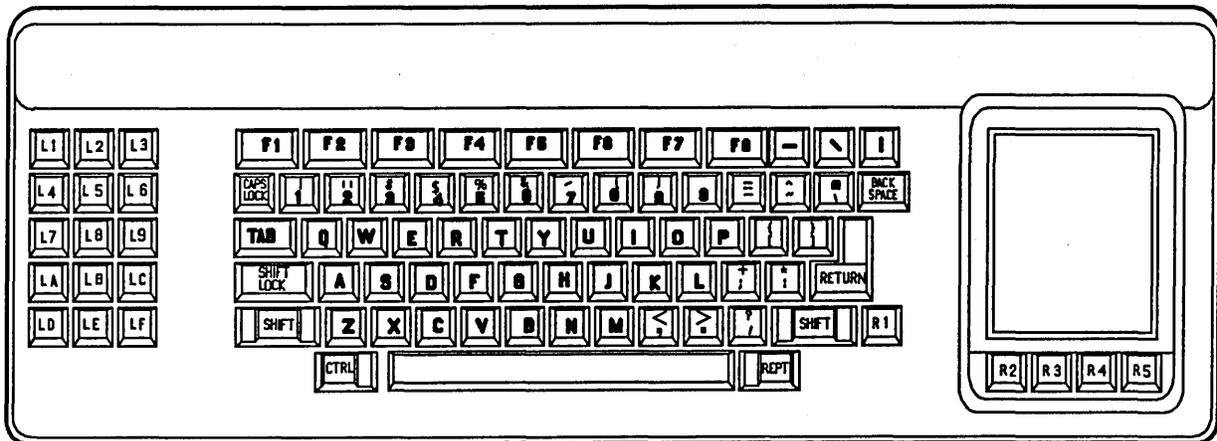


Figure 1-4. The 880 Keyboard Map

### 1.5.2. Controlling Keys from Within a Program

Because of the great flexibility provided by our displays and keyboards, many applications programs assume control of these and redefine various capabilities. When this happens, the default DM key definitions are overridden by the applications program. The default definitions are restored once the applications program ends. For your own applications, you may control key definitions through program calls to the PAD\_\$DEF\_PFK and PAD\_\$DM\_CMD routines as described in the *DOMAIN System Call Reference*.

Because the normal functions of the Display Manager keys are often useful (even when they have been redefined by applications programs), the <HOLD> and <HOLD/GO> keys have been defined to provide a temporary override function. Pressing <HOLD> while in an applications program will restore the keyboard to its log in DM definitions. Pressing <HOLD> again will re-enable the application-defined keys.

You may not change this feature of the <HOLD> and <HOLD/GO> keys, which is functional only when the keyboard is under applications program control. This capability is independent of the default Display Manager definitions of WH (WINDOW\_HOLD).

### 1.6. Special Characters in DM Scripts and Key Definitions

Several rules governing the use of literal and special characters affect the proper interpretation of commands within the Display Manager environment. The following characters have special meanings when they appear in a DM command line or script.

- @ The escape character "@" always nullifies any special meaning that the following character might have. As a part of command parsing, the Display Manager strips off the "@" character itself. If you can't remember whether a character has some special meaning to the Display Manager, it is always safe to escape the character -- if it is not special, the Display Manager still removes the "@", so the character appears as it should. The need for character escaping is generally confined to search and substitute operations, commands requiring quoted strings, and key definitions.

The use of "@" can be confusing in key definitions because the text in key definitions is actually processed twice - once when the definition is made, and once when the key is depressed and the definition is used. If a character needs to be escaped *both* times, it must be preceded by *three* "@" signs. For example, "@@@" becomes "@#" in the key definition, which then becomes "#" when the definition is used. Only the characters listed in this section are special within key definitions.

- # When read from a DM script, (via the CMDF command), the "#" character causes the remainder of the line to be treated as a comment and skipped.
- ; Semicolon is the normal command delimiter. It is equivalent to NEWLINE (generated by <RETURN>).
- & This character makes an input request, except when it is read from the keyboard. When read from the keyboard, it can be used in the replacement part of a substitution command to represent the entire string matching the regular expression. When "&" is preceded by "@" it becomes an ordinary character in both contexts. Therefore, you can't use "&" within a script or key definition and also use its special meaning within substitute commands that appear in that script or definition.

Some commands accept strings surrounded by single quotes. They are KD, ES, CP, CPO, CPS, and the "&" character. When you use single quotes, the only characters in the quoted string that retain their special meanings are "@", "&", and the closing single quote. All other characters revert to their literal graphic values. Note, however, that the KD command is not aware of single quotes within the definition string, so "#" and ";" must be quoted there as well.

For example, to define the F4 key to enter the string "-#-" at the current cursor position, place the following line in a key definition file:

```
kd F4 es '-@@@#-' ke
```

# Display Manager Task Lists

## Moving the Cursor

Task	DM Command	Predefined Key			
		Low-Profile		880 Keyboard	
Move left one character	AL	←	(LA)	←	(LA)
Move right one character	AR	→	(LC)	→	(LC)
Move up one line	AU	↑	(L8)	↑	(L8)
Move down one line	AD	↓	(LE)	↓	(LE)
Set arrow key scale factors (raster units)	AS x y	none		none	
Move to start of next line	AD;TL	CTRL/K		CTRL/K	
Move to beginning of line	TL	←	(L7)	←	(L4)
Move to end of line	TR	→	(L9)	→	(L6)
Move to top line in window	TT	<SHIFT>/↑	(LDS)	none	
Move to bottom line in window	TB	<SHIFT>/↓	(LFS)	none	
Move to window border	TWB {-L, -R, -T, -B}	none		none	
Tab left	THL	CTRL/<TAB>		CTRL/<TAB>	
Tab right	TH	<TAB>		<TAB>	
Set tabs	TS [n1 n2 ...] [-R]	none		none	
Move to DM Input Pad	TDM	<CMD>	(L5)	<CMD>	(L5)
Move to next window on screen	TN	<NEXT WNDW>	(LB)	<NEXT WNDW>	(LB)
Move to previous window	TLW	CTRL/L		CTRL/L	
Move to next window in which input is enabled	TI	none		none	
Move cursor to next unobscured icon on the display	TNI	none		none	
Control window availability for cursor placement	CURS [-ON -OFF]	none		none	

## Creating a Process

Task	DM Command	Predefined Key			
		Low-Profile		880 Keyboard	
Create new process, transcript pad, and windows	CP pathname	<SHELL>	(L5S)	<SHELL>	(R2)
Create new process without transcript pad or windows	CPO pathname	none		none	
Create server process (valid only in DM startup file)	CPS pathname	none		none	

## Controlling a Process

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Quit, stop, or blast a process	DQ [name] [-S -B -C nn]	CTRL/Q	CTRL/Q
Suspend execution of a process	DS	none	none
Resume execution of suspended process	DC	none	none

## Creating a Window Read and Edit Pads

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Create edit pad and window in which to view it	CE pathname	<EDIT> (R4)	<EDIT> (R4)
Create view window (read-only pad)	CV pathname	<READ> (R3)	<READ> (R3)
Create copy of existing window	CC	none	none

## Managing Windows

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Grow or shrink a window using rubberbanding	WGE	<GROW>	(L3A) CTRL/G
Move a window using rubberbanding	WME	<MOVE>	(L3AS) CTRL/W
Push or pop a window on stack	WP	<POP>	(R1) CTRL/P
Close (delete) a window	WC [name] [-Q   -F]	none	none
Close window, pad; update file	PW;WC -Q	<EXIT>	(R5) CTRL/Y
Close window, pad; no update	WC -Q	<ABORT>	(R5S) CTRL/N
Set scroll mode	WS [-ON   -OFF]	CTRL/S	CTRL/S
Set autohold mode	WA [-ON   -OFF]	none	none
Toggle scroll and autohold modes	WA;WS	CTRL/A	CTRL/A
Set hold mode	WH [-ON   -OFF]	<HOLD>	(R6) <HOLD/GO> (R5)
Set insert mode (see "Editing a Pad")			
Define position of default window n	WDF [n]	none	none
View latest output in Shell's process transcript pad	AU;AU;PB;TI	CTRL/V	CTRL/V
Copy text to Shell's process input pad	DR;TR;XC;TL;TI;TB; TR;XP;TR	<AGAIN>	(R2) <F8>
Copy text to DM input pad for use with <READ>	DR; /[-a-z0-9\$@/@-._`]/ XC CV_FILE;TDM; ES 'CV ' ;XP CV_FILE; TR;EN	<SHIFT>/<READ>	(R3S) none

## Moving a Pad Under a Window

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Move top of pad into window	PT	none	none
Move cursor to first character in pad	PT;TT;TL	CTRL/T	CTRL/T
Move bottom of pad into window	PB	none	none
Move cursor to last character in pad	PB;TB;TR	CTRL/B	CTRL/B
Move pad n pages	PP [-]n	↑ (LD, LF)	↓ ↑ (LD, LF)
Move pad n lines	PV [-]n	<SHIFT>/↑ (L8S)	<SHIFT>/↓ (LES) F2, F3
Move pad n characters	PH [-]n	<SHIFT>/← (LAS)	<SHIFT>/→ (LCS) ← → (L7, L9)
Save pad in pathname	PN pathname	none	none

## Editing a Pad

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Set read/write mode	RO [-ON   -OFF]	CTRL/M	CTRL/M
Set insert/overstrike mode	EI [-ON   -OFF]	<INS> (L1S)	<INS MODE> (L1)
Insert string	ES 'string'	[Default DM operation]	
Insert NEWLINE character	EN	<RETURN>	<RETURN>
Insert new line after current line	TR;EN;TL	<F1>	<F1>
Insert raw (noecho) character	ER nn	none	none
Insert end-of-file mark	EEF	CTRL/Z	CTRL/Z
Delete character at cursor	ED	<CHAR DEL> (L3)	<CHAR DEL> (L3)
Delete character before cursor	EE	<BACK SPACE>	<BACK SPACE>
Delete word of text	DR;[:-A-Z0-9\$_]/XD	<F6>	<F6>
Delete from cursor to end of line	ES ' ';EE;DR;TR;XD;TL;TR	<F7>	<F7>
Delete entire line	CMS;TL;XD	<LINE DEL> (L2)	<LINE DEL> (L2)
Copy text to paste buffer	XC [-F pathname   name]	<COPY> (L1A)	CTRL/C
Cut (delete) text and write it to paste buffer	XD [-F pathname   name]	<CUT> (L1AS)	CTRL/E
Paste (write) text in paste buffer into pad	XP [-F pathname   name]	<PASTE> (L2A)	CTRL/O
Copy a portion of the screen into a GMF file	XI [pathname]	none	none
Search forward for string	/string/	none	none
Repeat last forward search	//	CTRL/R	CTRL/R
Search backward for string	\string\	none	none
Repeat last backward search	\\	CTRL/U	CTRL/U
Abort search	ABRT (or) SQ	CTRL/X	CTRL/X
Set case comparison for search	SC [-ON   -OFF]	none	none
Substitute string2 for all occurrences of string1 in defined range	S/string1/string2/	none	none
Substitute string2 for first occurrence of string1 in each line of defined range	SO/string1/string2/	none	none
Check and/or balance delimiting characters	BL [-I -C] [l_char] [r_char]	none	none
Undo previous command(s)	UNDO	<UNDO> (L2AS)	none
Update edit file without closing edit pad	PW	<SAVE> (R4S)	none

## Using Window Groups and Icons

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Add a window or group to a new or existing group of windows	WGRA group_name [entry_name]	none	none
Remove a window or group from a group of windows	WGRR group_name [entry_name]	none	none
Control window visibility	WI [entry_name][-I   -W]	none	none
Change a window into an icon or an icon into a window	ICON [entry_name][-I   -W] [-C 'char']	none	none
Set icon position and offset vector	IDF	none	none
Display a list of the windows in a group	CPB	none	none

## Managing the Display Environment

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Request help	none	<HELP> (R6S)	none
Log in	L id [proj [org]] [-P] [-H]	none	none
Log off	LO [-F] [-ON -OFF]	none	none
Shut down system	SHUT [-F]	none	none
Place a mark	DR	<MARK> (L1)	<MARK> (R1)
Go to a mark	GM	none	none
Clear mark stack	CMS	none	none
Display cursor coordinates (line, column; x/y coord.)	=	none	none
Acknowledge DM alarm	AA	none	none
Acknowledge alarm and pop window	AP	none	none
Set or display an environment variable	ENV var [value]	none	none
Set background color	BGC [-ON   -OFF]	none	none
Set window color	INV [-ON   -OFF]	none	none
Set color monitor to black & white	MONO [-ON   -OFF]	none	none
Refresh entire screen	RS	CTRL/F	CTRL/F
Refresh window	RW [-R]	none	none
Load font for use in pads	FL pathname	none	none
Declare keyboard type	KBD [type]	none	none
Set or display key definition	KD name [[def.] KE]	none	none
Request input to command line	&'prompt'	none	none
Execute DM script	CMDF pathname	none	none



**Chapter 2**  
**Display Manager Commands**

AA (ACKNOWLEDGE\_ALARM)

**AA (ACKNOWLEDGE\_ALARM) -- Acknowledge Display Manager Alarms**

**FORMAT**

**AA**

The AA command acknowledges a Display Manager alarm. This command turns off the current alarm and enables further alarms, which may already be waiting. AA requires no arguments or options.

**ABRT (ABORT) -- Abort text search; cancel any action involving ECHO**

**FORMAT**

**ABRT (CTRL/X)**

The ABRT command aborts a text search, and cancels any action involving the ECHO command.

When you use ABRT to abort the current search, the Display Manager returns the message "Search aborted." It does not move the window. Note that you cannot type this command during a search.

When you use ABRT to abort the ECHO command, ABRT cancels a move window with rubberbanding or grow window with rubberbanding operation; or, it cancels highlighting for a defined range of text, depending on how ECHO was used.

CTRL/X issues this command by default.

AD (ARROW\_DOWN)

AD (ARROW\_DOWN) -- Move cursor down one line.

## FORMAT

### AD

The AD command moves the cursor down one line from its current position. By default, the down arrow key (LE) on the left hand key pad executes this command.

**AL (ARROW\_LEFT) -- Move cursor left one character.**

**FORMAT**

**AL**

The AL command moves the cursor left one character from its current position. By default, the left arrow key (LA) on the left hand key pad executes this command.

AP (ACKNOWLEDGE\_POP)

AP (ACKNOWLEDGE\_POP) -- Acknowledge alarm and pop window.

## FORMAT

### AP

The AP command acknowledges a Display Manager alarm and displays ("pops") the window to which the alarm pertains. This command is particularly useful if the window is completely covered so that you cannot point to it. AP requires no arguments or options.

**AR (ARROW\_RIGHT) -- Move cursor right one character.**

**FORMAT**

**AR**

The AR command moves the cursor right one character from its current position. By default, the right arrow key (LC) on the left hand key pad executes this command.

AS (ARROW\_SCALE)

AS (ARROW\_SCALE) -- Set scale factors for arrow keys.

## FORMAT

AS [x[y]]

The AS command sets scale factors for the arrow keys. The scale factor is useful for changing the apparent sensitivity of the arrow keys and for lining up the edges of windows after moving them.

## ARGUMENTS

If no arguments are specified, then the default scale factors are used, as described below.

**x**

(optional)

Specify horizontal scale factor in raster units (integer). This value must be in the range 0-1023. (Note, however, that portrait displays may only display up to 800 raster units in this dimension.) There are approximately 100 raster units per inch. The default horizontal movement is the width of the character on which the cursor rests; if the cursor is not on a character, the DM uses the width of a space in the last window. Specifying 0 for 'x' indicates that the default should be used.

Default if omitted: 0

**y**

(optional)

Specify vertical scale factor in raster units (integer). This value must be in the range 0-1023. (Note, however, that landscape displays may only display up to 800 raster units in this dimension.) The default vertical movement is the height of a line in the last window. Specifying 0 for 'y' indicates that the default should be used.

Default if omitted: leave current y value unchanged

AU (ARROW\_UP) -- Move cursor up one line.

**FORMAT**

**AU**

The AU command moves the cursor up one line from its current position. By default, the up arrow key (L8) on the left hand key pad executes this command.



BGC (BACKGROUND\_COLOR)

BGC (BACKGROUND\_COLOR) -- Set background color of display.

## FORMAT

**BGC [-ON | -OFF]**

The BGC command sets the background color for monochrome displays. Note that this is the *display* background only; window background color is controlled with INV (INVERT\_COLOR).

The background color is ON, by default, at login.

NOTE: BGC has meaning only for monochromatic displays. It has no effect on nodes with color displays. See the DM command MONO for information about background color on color displays.

## OPTIONS

If no option is specified, BGC toggles the current mode.

- ON**                      Set background color to grey or green, depending on display type.
- OFF**                     Set background color to black.

**BL (BALANCE) -- Balance delimiters.**

## FORMAT

**[range]BL [-I | -C] [l\_char] [r\_char]**

BL determines whether a given pair of delimiting characters (for instance, left and right parentheses) is balanced within a specified range of text.

## ARGUMENTS

**range**

**(optional)**

Specify range of text to be checked. This argument is valid only when used with -C; the range for -I is the current cursor position to the end (or beginning) of the file. Define the range to be checked as described in "Defining a Range of Text" in the previous chapter.

Default if omitted: check from cursor to end of line

**l\_char**

**(optional)**

Specify the left delimiting character. If 'r\_char' is specified but this argument is omitted, the left delimiting character defaults to 'r\_char'. If both arguments are omitted, the left delimiting character defaults to left parenthesis.

**r\_char**

**(optional)**

Specify the right delimiting character. If 'l\_char' is specified but this argument is omitted, the right delimiting character defaults to 'l\_char'. If both arguments are omitted, the right delimiting character defaults to right parenthesis.

## OPTIONS

If either of the following options is specified, it must precede any specified arguments.

Default options are indicated by "(D)."

**-I**

**(D)**

Insert mode: search for balanced delimiters from the current character to the beginning or end of the file. The behavior of BL depends upon the character under the cursor when BL is invoked.

If you position the cursor on a delimiter and BL finds a balancing delimiter, it moves the cursor to the matched character momentarily (to show you where the balance is completed), then returns the cursor to the character immediately following the initial cursor position. The search direction is forward if the character under the cursor is a left delimiter, and backward if the character is a right delimiter.

If you position the cursor on a delimiter and BL finds no

BL (BALANCE)

balancing delimiter, it gives an error message and sounds the alarm, then inserts a matching right delimiter at the initial cursor position.

If you position the cursor on a character other than a delimiter, BL searches backward for the first occurrence of 'l\_char', briefly shows you where it is, then inserts a matching right delimiter at the initial cursor position.

-C

Check mode: check only -- do not insert balancing characters or move the cursor. You may mark a range of text to be checked if you specify this option; see the 'range' argument above. BL checks all pairs of specified delimiters within the specified range. Results of the check are displayed in the DM message window.

**CASE -- Change case of all letters in a defined range of text.**

### FORMAT

**[range]CASE [options]**

The CASE command changes the case of all the letters in a defined range of text. You can instruct CASE to invert the case of all letters, change all letters to uppercase, or change all letters to lower case. If you do not specify a range, CASE operates on the text from the cursor position to the end of the current line.

### OPTIONS

Default options are indicated by "(D)."

- S (D)** Swap all uppercase letters for lowercase, and all lowercase letters for upper case (in the defined range).
- U** Change all letters in the defined range to uppercase.
- L** Change all letters in the defined range to lowercase.

CC (CREATE\_COPY)

CC (CREATE\_COPY) -- Create a copy of an existing window.

## FORMAT

### CC

The CC command creates a copy of an existing window. To use this command, mark both edges of the new window with the DR command, then point to the window to be copied. With the cursor in the window to be copied, press <CMD> and issue the CC command.

If no region is marked, the new window is created using the next default DM window.

### \*NOTE\*

There is a homonymous Shell command: CC (COMPLE\_C) -- Compile a C program. See the CC command description in the *DOMAIN C Language Reference* for more information.

**CDM (CHANGE\_DISPLAY\_MANAGER\_MODE) -- Change the display manager mode.**

## FORMAT

**CDM [-P 1 | 8]**

The CDM command changes the display mode of the hardware which affects the colors used by the Display Manager. A user normally uses this command in preparation for running a direct color application, which requires a 24-plane workstation. In this case it is necessary to restrict the Display Manager to only using 2 colors.

At login, the default is CDM (with no options), which instructs the hardware to use the highest number of planes (normally 8) when drawing colors. This is an indirect color mode where the DM uses several colors for window banners, window background, and text.

Note that this command causes a visual change in the colors on the screen of a 24-plane workstation only. It has no effect on any other display hardware and the DM will give an error message, "wrong display hardware", if this command is issued on any device other than a 24-plane workstation.

The CDM command differs from the MONO command in that the MONO command does not affect the 24-plane hardware in any way. The MONO command simply instructs the DM to use black and white for all its drawing operations, thus freeing up color slots in the color map.

## OPTIONS

The only option that the CDM command takes is a -P option that allows the user to specify the number of planes that the Display Manager should use to get color. For example, a "CDM -P 1" causes all DM output to be displayed in only 2 colors, through the use of one plane. This is necessary to "free up" all 24 planes so that some application can run in direct color mode. When the user is finished running a direct color application, the DM can be restored to its original state by issuing the "CDM -P 8" command. The DM's original state is such that it uses 4 colors for window background, 4 more colors for window banner background, white for banner text, and black for text in DM windows.

If no options are specified, CDM defaults to highest number of planes, causing the the display to be reset to its original state where existing indirect color applications work as before.

Default options are indicated by "(D)."

- P 1**                      This causes the Display Manager to put the hardware in a state where the DM draws in only 1 plane, causing the DM's output to appear in 2 colors.
- P 8**                      **(D)**      This instructs the DM to use all 8 planes for drawing. The hardware mode is changed to allow the DM to use 8 planes. The DM's output appears in many colors. This option is currently equivalent to giving the CDM command with no options.



**CE (CREATE\_EDIT) -- Create an edit pad and window.**

## FORMAT

**[region]CE pathname [options]**

Giving the CE command causes the Display Manager to create an edit pad and a window in which to view it. If the file specified exists, it is opened for editing. If the file does not exist, the DM creates and opens a file with the specified name.

By default, the <EDIT> key (R4) invokes the CE command, automatically moving the cursor to the DM input pad and issuing the "Edit file: " prompt. Type the pathname of the file to be edited.

Once an edit pad is created, you may use other DM commands to manipulate text in the pad. See the DM index pages at the end of the previous chapter for a list of editing commands.

To close a pad and window, use the DM command WC (usually CTRL/N).

NOTE: CE does not create a process. It simply opens a file for editing within the current Display Manager process.

## ARGUMENTS

**region**  
(optional)

Specify area of the screen where the new window will be displayed. For details on window boundaries, see the previous chapter.

Default if omitted: use next DM default window.

**pathname**  
(required)

Specify file to be edited.

## OPTIONS

**-I**

Specify that the window created for this pad will be in icon format.

**-C 'char'**

Specify the icon character to be used in the icon window. 'char' must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the default icon character for this pad type.

C MDF (COMMAND \_FILE)

C MDF (COMMAND \_FILE) -- Execute DM script

## FORMAT

**C MDF pathname**

C MDF directs the Display Manager to read commands from a file (DM script). When it reaches the end of the file, the cursor returns to its previous location.

Command files may be nested; i.e., C MDF may be used within another DM script.

## ARGUMENTS

**pathname**  
**(required)**

Specify name of file to be executed. DM commands may appear one per line, or several per line, each delimited by semicolons.

**CMS (CLEAR\_MARK\_STACK) -- Erase existing marks.**

**FORMAT**

**CMS**

The CMS command erases any existing marks. Use it to ensure that commands requiring marked regions do not behave unexpectedly as a result of outstanding (but probably forgotten) marks. The <LINE DEL> standard key definition is a good example: "CMS;TL;XD". Previous marks are cleared so that only the current line is deleted.

CMS requires no arguments or options.

CP (CREATE\_PROCESS)

CP (CREATE\_PROCESS) -- Create process, pads, and windows.

## FORMAT

[region]CP [options] pathname [args ...]

CP creates a process, input and transcript pads, and input and transcript windows, then executes a program (indicated by the pathname argument) within that process. The transcript pad is opened as standard output in the new process. To create a process running the Shell that we supply, either press the <SHELL> key (which causes everything to happen automatically) or specify /COM/SH for pathname as follows:

Command: CP /COM/SH <RETURN>

This command creates an input pad associated with the transcript pad, and opens it as standard input. As a result, you have a new process, windows to its input and transcript pads, and a shell.

To stop a shell and delete all windows and pads associated with its parent process, type CTRL/Z in the shell's process input pad, then CTRL/N (refer to the DQ command in the next section for more information).

## ARGUMENTS

**region**  
(optional)

Specify area of the screen where the new window will be displayed. For details on window regions, see the previous chapter.

Default if omitted: use next DM default window.

**pathname**  
(required)

Specify file to be executed by the new process: usually a Shell (command interpreter).

**args ...**  
(optional)

Specify any arguments to be passed to the program 'pathname'. If any of these arguments contain explicit blanks, enclose those arguments in quotation marks.

## OPTIONS

Note that options, if present, must precede the 'pathname' argument.

**-I**

Specify that the window created for this process will be in icon format.

**-C 'char'**

Specify the icon character to be used in the icon window. 'char' must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the default icon character for this pad type.

**-N name**

Assign process name 'name.' If omitted, the DM assigns the name "Process\_n," where n is an integer beginning with 1 and incremented by 1 for each active process.

### EXAMPLES

1. Create a process named 'spare' running the Shell. The '-nstart' option on SH suppresses startup file execution for the new Shell.

Command: (0,0)dr;(500,300)cp /com/sh -nstart -n spare

2. Create a process running the Shell, and place it in a window in icon format using the default icon character for this pad type.

Command: cp -i /com/sh

CPB (CREATE\_PASTE\_BUFFER)

CPB (CREATE\_PASTE\_BUFFER) -- Display a list of the windows in a group.

## FORMAT

CPB **group\_name** [options]

The CPB command creates a window on a named paste buffer specific to the given group. The paste buffer contains a list of the windows in the group. Because these group lists are held in paste buffers, your programs can access the groups by using the PBUFS routines described in the *DOMAIN System Call Reference*.

The DM automatically creates three special paste buffers to help you manage your windows and icons. These paste buffers contain the following groups:

- The INVIS\_GROUP -- this buffer holds the pathnames of all the windows that you have made invisible.
- The ICON\_GROUP -- this buffer holds the pathnames of all the windows that are represented by icons.
- The ALL\_GROUP -- this buffer holds the pathnames of every window open on your node - Shell process windows, DM windows, visible and invisible windows, and windows represented by icons.

These special groups are created regardless of any other groups, and their members may overlap with the members of any other group (just as any group can have the same member(s) as another).

A special feature of the CPB command allows you to directly access the windows in a group when the paste buffer holding the group is displayed on your screen. To use this feature:

1. Use the CPB command to display the list of windows.
2. Position the cursor on the pathname of the window you want to access.
3. Press <CMD>, and issue the DR (MARK) command.
4. Press <CMD> again, and issue the desired DM command.

By using this feature you can directly access windows that are invisible, represented with icons, etc.

## ARGUMENTS

**group\_name**  
(required)

Specify the name of the group you want to display.

## OPTIONS

**-I**

Specify that the window created will be in icon format.

**-C 'char'**

Specify the icon character to be used in the icon window. 'char' must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the default icon character for this pad type.

CPO (CREATE\_PROCESS\_ONLY)

CPO (CREATE\_PROCESS\_ONLY) -- Create process without pads or windows.

## FORMAT

**CPO [options] pathname [args...]**

The CPO command creates only a process, without associated pads or windows. The four standard I/O streams are directed to /DEV/NULL. If this command appears in the node's DM boot startup script 'NODE\_DATA/STARTUP the system assigns the new process the subject identifier (SID) USER.SERVER.NONE.local\_node, and the created process will continue to run regardless of whether or not any one is logged in. This is desirable for utilities like the PRSVR (PRINT\_SERVER) and NETMAN, and means that CPO is identical to CPS in this context.

If CPO is issued in any other startup script or from the keyboard, the SID of the new process is derived from whatever process invokes CPO, and the created process will terminate at logout.

## ARGUMENTS

**pathname**  
(required)

Specify file to be executed by the new process.

**args...**  
(optional)

Specify any arguments to be passed to the program 'pathname'. If any of these arguments contain explicit blanks, enclose those arguments in quotation marks.

Default if omitted: no arguments passed.

## OPTIONS

**-N name**

Assign process name 'name'. If this option is omitted, the process is not named.

**-W**

Invoke "wait" mode. If this option is specified, the DM suspends its activities until the newly created process terminates. As long as the process runs, the DM will not respond to keyboard or other input. **USE THIS OPTION WITH CAUTION.** If the newly created process does not terminate, the DM will appear to be hung. In addition, processes created using -W CANNOT make any DM requests (via pad\_\$ requests or DM commands) because the DM is suspended and will not respond.

## EXAMPLES

1. Run the ALARM\_SERVER in a background process.

Command: `cpo /sys/alarm/alarm_server -disk 98 -bell1`

**CPS (CREATE\_PROCESS\_SERVER) -- Create process independent of login.**

## FORMAT

**CPS [options] pathname [args...]**

CPS creates a process (without associated pads or windows) that runs regardless of whether or not any one is logged in. This is desirable for utilities like the PRSVR (PRINT\_SERVER) and NETMAN. CPS may appear in any of the DM startup scripts. You may prefer to issue the CPS command from the keyboard on selected occasions, however, rather than include this function in a startup script.

The created process is assigned the subject identifier (SID) USER.SERVER.NONE.local\_node regardless of the context in which the CPS command appears. Be sure that any files to be used by this process (including the program specified by the 'pathname' argument) give adequate access to this SID. If the ACLs on the files do not allow proper access to the SERVER project name, the process will terminate. And since background processes are essentially invisible, no error messages are returned to the display, making fault diagnosis difficult.

## ARGUMENTS

**pathname**  
(required)

Specify file to be executed by the new process.

**args...**  
(optional)

Specify any arguments to be passed to the program 'pathname'. If any of these arguments contain explicit blanks, enclose those arguments in quotation marks.

Default if omitted: no arguments passed.

## OPTIONS

**-N name**

Assign process name 'name'. If this option is omitted, the process is not named.

**-W**

Invoke "wait" mode. If this option is specified, the DM suspends its activities until the newly created process terminates. As long as the process runs, the DM will not respond to keyboard or other input. **USE THIS OPTION WITH CAUTION.** If the newly created process does not terminate, the DM will appear to be hung. In addition, processes created using -W CANNOT make any DM requests (via pad\_\$ requests or DM commands) because the DM is suspended and will not respond.

## EXAMPLES

1. Run the server MBX\_HELPER.

Command: `cps /sys/mbx/mbx_helper -n mbx_helper`

CURS (CURSOR\_LOCK)

CURS (CURSOR\_LOCK) -- Control cursor positioning.

## FORMAT

CURS [-ON | -OFF]

CURS controls whether or not a window is available for cursor positioning by the DM command TN (TO\_NEXT\_WINDOW), normally invoked by <NEXT WNDW>. All windows initially default to CURS -ON, which permits the DM to move the cursor into all windows via the TN command.

CURS operates on a per-window, per-pane basis. For example, you may prevent the DM from moving the cursor to a transcript pad's pane while permitting it to move the cursor to the related input pad's pane.

To set the window state, simply point to the appropriate window and then issue the CURS command.

## OPTIONS

If no option is specified, CURS toggles the current mode.

- ON                      Enable cursor positioning.
- OFF                     Disable cursor positioning.

**CV (CREATE\_VIEW) -- Create a read-only edit pad and window.**

## FORMAT

**[region] CV pathname [options]**

The CV command creates a read-only edit pad to view an existing file. You may not make changes to the file, only view it. If you decide that you want to make changes to it after all, you must first disable read-only mode. See the RO command description for details about that operation.

By default, the <READ> key (R3) invokes the CV command, automatically moving the cursor to the DM input pad and issuing the "Read file: " prompt. Type the pathname of the file to be read.

To close a pad and window, use the DM command WC (usually CTRL/N).

## ARGUMENTS

**region**  
(optional)

Specify area of the screen where the new window will be displayed. For details on window regions, see the previous chapter.

Default if omitted: use next DM default window.

**pathname**  
(required)

Specify file to be viewed. An error occurs if the file does not exist.

## OPTIONS

**-I**

Specify that the window created for this pad will be in icon format.

**-C 'char'**

Specify the icon character to be used in the icon window. 'char' must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the default icon character for this pad type.

DC (DEBUG\_ CONTIUNE)

DC (DEBUG\_ CONTINUE) -- Continue a suspended process.

## FORMAT

### DC

The DC command restarts a process that has been suspended by the DS (DEBUG\_ SUSPEND) command. Refer to the DS command description for details about that operation.

DC requires no arguments or options.

**DQ (DEBUG \_QUIT) -- Generate a quit fault in a process.**

## FORMAT

**DQ [entry\_name] [options]**

The DQ command generates a quit fault, which normally interrupts execution of the current program and returns the process to the calling program (usually the Shell). This command affects the process associated with the window that contains the cursor. By default, CTRL/Q invokes DQ without options to generate a normal quit fault in the program currently running.

## ARGUMENTS

**entry\_name**  
(optional)

Specify the name of the window or window group whose process is to receive the fault. Note that this is valid only for processes with windows. To stop background processes, use the Shell command SIGP (SIGNAL\_PROCESS). If the name of the window or group appears as a text string somewhere on the display, you may use the following time-saving feature: place the cursor on the name, then press <MARK>. Now issue the DQ command. DQ uses the MARKed name for the 'entry\_name' argument.

Default if omitted: send fault to the process whose window is under the cursor.

## OPTIONS

If no options are specified, then DQ generates a normal quit fault and halts whatever program is currently running.

- C nn**                   Generate an arbitrary asynchronous fault with the specified hexadecimal status (nn).
- S**                       Stop entire process in a controlled way, if possible. Close open streams, files, pads, etc. The Shell's parent process is stopped and closed, too.
- B**                       Blast process; do not execute further user mode instructions.

**NOTE:** If you are trying to stop a Shell's parent process (as opposed to some program running within a Shell), there is an easier method than typing DQ -S. Position the cursor in the Shell's process input window and issue an EOF (END\_OF\_FILE) command (by default, CTRL/Z). This signals completion of input, and stops both the Shell and the process. See the description of the DM command EOF for more information.

DR (DEFINE\_REGION)

DR (DEFINE\_REGION) -- Place a mark to define a region.

## FORMAT

### DR

The DR command marks some part of the display or some part of a pad. The mark can be used to define a region for a substitute command, to grow, shrink, or move a window, or to reposition the cursor.

You may specify a literal point at which the mark is to be placed by preceeding the DR command with line and column numbers in a pad, x and y screen coordinates, or regular expressions for matching text. If no point is specified, the mark is placed at the current cursor position.

By default, the <MARK> key invokes the DR command along with ECHO to provide user-visible feedback.

**DS (DEBUG\_SUSPEND) -- Suspend a process.**

**FORMAT**

**DS**

The DS command generates a temporary interrupt for a process. All activities are suspended. Processes may be restarted with the DC (DEBUG\_CONTINUE) command. See the DC command description for details.

DS requires no arguments or options.

## ECHO

**ECHO -- Begin text echoing, end rubberbanding.**

## FORMAT

**ECHO [option]**

When used as part of the DR; ECHO command sequence invoked with <MARK>, the ECHO command performs two separate operations depending on the situation where it is used. When you press <MARK> to begin defining a range of text, ECHO tells the DM to begin highlighting the indicated text range in reverse video (text echoing). When you use <MARK> to complete a move window (WME) or grow window (WGE) operation, the ECHO command tells the DM to remove the "rubberband" and move or grow the window as indicated. You can abort text highlighting or rubberbanding using the SQ command. (By default, the CTRL/X key combination issues the SQ command.)

## OPTIONS

**-R**

Specify ECHO for a rectangular region of text. Use a mark point and the cursor to specify a column along the left side of the text you want to highlight in reverse video. When you issue the ECHO command with the -R option all text to the right of the specified column is then displayed in reverse video.

**ED (EDIT\_DELETE) -- Delete character under cursor.**

**FORMAT**

**ED**

The ED command deletes the character under the cursor. If the character is a NEWLINE, ED joins two lines. By default, the <CHAR DEL> key invokes the ED command.

ED requires no arguments or options.

**NOTE:** There is a homonymous Shell command: ED -- Invoke line mode editor. See the ED command description in the Shell chapter for details.

EE (EDIT\_ERASE)

EE (EDIT\_ERASE) -- Delete character preceding cursor.

#### FORMAT

##### EE

The EE command deletes the character preceding the cursor. If the window is in overstrike mode, EE replaces the preceding character with a blank. By default, the <BACKSPACE> key invokes the EE command.

EE requires no arguments or options.

**EEF (EDIT\_END\_OF\_FILE) -- Insert end-of-file mark.**

## **FORMAT**

### **EEF**

The EEF command inserts a stream end-of-file mark (EOF) in the pad. If the line containing the cursor is empty, the end-of-file mark is written on that line. Otherwise, the end-of-file mark is inserted following the current line.

By default, CTRL/Z executes the EEF command.

It is a common (although not universal) convention for programs to terminate execution and return to the process which called them when they receive an EOF on their standard input stream. The command Shell is such a program. When the top-level program in a process (usually /COM/SH) returns, the process stops and all of its streams are closed. The Display Manager then closes the Shell's process input pad and window, and closes the transcript pad. Whether or not the transcript window also disappears depends on the setting of its auto-close mode (see the WC command description for details). If auto-close is disabled (the default condition), then you must manually delete any windows associated with the closed transcript pad by using the DM command WC -Q, or CTRL/N.

EI (EDIT\_INSERT)

EI (EDIT\_INSERT) -- Set insert/overstrike mode.

## FORMAT

EI [-ON | -OFF]

The EI command puts the current pad into (-ON) or out of (-OFF) insert mode. If no option is supplied, the current mode is inverted. In insert mode, characters you type are inserted into the pad without replacing or overstriking any existing characters. This causes existing text to drift to the right as new text is added. In overstrike mode (i.e., insert mode turned off), characters typed at the keyboard replace those under the cursor. This can be useful for entering information into pre-formatted files so that the format is undisturbed.

By default, the <INS> key on low-profile keyboards and the <INS MODE> key on 880 keyboards invoke the EI command without options to toggle the current mode.

The window legend contains an "I" when the window is in insert mode. The "I" disappears in overstrike mode.

All pads are initially in insert mode, although this is irrelevant if the pad is also read-only.

EN (EDIT\_NEWLINE) -- Insert NEWLINE.

**FORMAT**

**EN**

The EN command inserts (or overstrikes, depending on current mode) a NEWLINE character at the current cursor position.

By default, the <RETURN> key invokes this command.

## ENV (ENVIRONMENT)

**ENV (ENVIRONMENT) -- Set or display an environment variable.**

### FORMAT

**ENV variable [value]**

The DM command ENV sets or displays the value of an environment variable. Environment variables are of primary concern to DOMAIN/IX users; please consult the DOMAIN/IX documentation for details about their usage.

If you invoke ENV from the keyboard, you may use it only to **display** environment variables, not set them. To set variables, ENV must appear in one of your startup scripts so that it gets executed before any Shells are created. This is because the DM assigns values to environment variables for new Shells using those in effect for the window that currently contains the cursor. ENV thus does not have a chance to influence the new Shell if other Shell(s) already exist. In addition, the ENV command will NEVER change the value of a variable in an existing process.

For details about manipulating environment variables from the Shell, see the EXPORT command description in the Shell commands chapter.

### ARGUMENTS

**variable**  
**(required)**

Specify the name of the variable whose value is to be set or displayed. Since the DM normally forces arguments to uppercase prior to command scanning, enclose a variable whose name must be lowercase in single quotation marks.

**value**  
**(optional)**

Specify the new value to be assigned to 'variable'. Since the DM normally forces arguments to uppercase prior to command scanning, enclose a value which must be lowercase in single quotation marks.

Default if omitted: display the current value of 'variable'.

### EXAMPLES

Command: env SYSTYPE	Display the current value for SYSTYPE for the current Shell window.
Command: env SYSTYPE 'bsd4.2'	Set the SYSTYPE variable to 'bsd4.2'. This line must appear in a startup script to have any effect.

**ER (EDIT\_RAW) -- Insert raw character.**

#### **FORMAT**

##### **ER nn**

The ER command sends a raw character to a program. The single argument nn (required) is a one or two character hexadecimal value which defines the single byte sent to the active program the next time the program requests input. The data byte is not echoed anywhere on the display. In effect, this command delivers a single raw keystroke to a program.

This command can be used by programs that need to define keys to return known values for actions by the programs.

This command differs from the other text insertion commands in that it does NOT insert the hexadecimal character into an edit pad. Its sole function is to pass a hexadecimal character to a running program.

ES (EDIT\_STRING)

ES (EDIT\_STRING) -- Insert string.

## FORMAT

### ES 'string'

If a window is currently in write mode, then any text character typed at the keyboard is inserted at the current cursor position. This is the default Display Manager action. Typing text into a read-only window causes an error.

The ES command inserts a string of text at the current cursor position. Enclose the string to be inserted in apostrophes. Since text insertion is the default action anyway, this command is primarily useful in key definition commands where you want some text written out when the key is pressed, or in DM scripts for writing text to the display.

**EX (EXIT) -- Exit Display Manager to Boot Shell.**

### **FORMAT**

#### **EX**

The EX command causes the system to stop the Display Manager process and enter the Boot Shell. This puts you in the same place that you would be if you had powered up your node with the NORMAL/SERVICE switch set to SERVICE.

To restart the Display Manager, type

) GO

in the Boot Shell.

This command differs from SHUT, which shuts the node's operating system (AEGIS) down completely and enters the Mnemonic Debugger that resides in the node's boot PROM.

Do not confuse this command with the Shell command EXIT, which exits a Shell script loop. See the EXIT command description for more information.

**FL (FONT\_LOAD)**

**FL (FONT\_LOAD) -- Load a font for use in pads.**

## **FORMAT**

**FL pathname [-I]**

The FL command loads a font for use in subsequent pads. Note that fonts apply to pads, not windows, so any new window opened to an old pad uses the old font.

You can load up to 50 fonts. The DM keeps track of fonts loaded by the FL command or programs. It unloads fonts on a least-recently-used basis.

If you need to unload a font (to edit it with EDFONT for example), issue an FL command for another font, and close all the windows using the font you wish to unload. If a program loaded the font, stop the program (issue an end-of-file -- usually CTRL/Z), and close the window.

## **ARGUMENTS**

**pathname**  
**(required)**

Specify name of file containing font to be loaded. The given pathname is first looked up directly, using the user working and naming directory rules. If not found, it is looked up in the directory /SYS/DM/FONTS.

## **OPTIONS**

**-I**

Specify that the font to be loaded is an icon font.

GM (GO\_TO\_MARK) -- Go to a mark.

## FORMAT

### GM

The GM command repositions the cursor at the most recently marked point after first marking the current cursor position (where you invoked the GM command). This allows you to alternate between two points with repeated invocations of GM. The most common use of this is to "remember" a position in a file and return to it later. GM repositions the window, if necessary, to display the marked pad location.

Pads may contain multiple marks, in which case GM works its way progressively through the mark stack, consuming each mark as it goes. You may also place marks in different pads; GM builds only one mark stack, not one for each pad. Thus you may use this command to jump around between pads.

GM requires no arguments or options.

## ICON

**ICON -- Change a window or window group into an icon(s); change an icon back into a window.**

### FORMAT

**ICON [entry\_name] [options]**

The ICON command changes the specified window or group into an icon(s), or changes an icon back into a window. You can use two methods to change a window into an icon: specify the window name (shown in the window legend) when you issue the ICON command, (either by typing it or using <MARK> as described below), or simply position the cursor in the window, press <CMD>, and issue the ICON command. If you want to change a group of windows into icons, you must specify the group name when you issue the ICON command. By default, if you do not specify an entry\_name (the name of a window or group) ICON manipulates the window under the cursor.

To change an icon back into a window, repeat the process described above. The window reappears on your display at its former position.

When you change a window into an icon, the DM displays an icon character that describes the type of information the window displayed, such as an edit pad, a graphics file, or a Shell transcript pad. The default icon characters are held in a font file called /SYS/DM/FONTS/ICONS. If you desire, you can examine or change this file using the EDFONT program described in the appendices. If you want to use your own icon font file, invoke the FL (FONT\_LOAD) command with the -I option prior to issuing the ICON command.

### ARGUMENTS

**entry\_name**  
**(optional)**

Specify the name of the window or group you want to change into icon(s), or change back into a window. If the name of the window or group appears as a text string somewhere on the display, you may use the following time-saving feature: place the cursor on the name, then press <MARK>. Now issue the ICON command. ICON uses the MARKed name for the 'entry\_name' argument.

Default if omitted: manipulate the window under the cursor.

### OPTIONS

If no options are specified, ICON toggles the current window setting.

- I** Force the window or group to appear as an icon. Not valid if -W is specified.
- W** Force the window or group to appear as a window. Not valid if -I is specified.
- C char** Specify the the DM icon character used to represent a window.

**IDF (ICON\_DEFAULT) -- Set the icon default positioning and offset.**

## FORMAT

### IDF

The IDF command sets the position of an icon on your screen, determines where subsequent icons will be positioned (the offset), and specifies the icon shift vector to use when icons start to overlap each other. Each time you issue the IDF command you reset the positions where any subsequent icons appear.

By default, icons appear in a horizontal line along the top of portrait displays, and in a vertical line along the right side of landscape displays. The default offset is set at the width of one icon (60 bits) horizontally or vertically, depending on the display. You can use IDF to change this default positioning and offset, to establish the position of an icon created in a script, or to set your personal icon positioning and offset in a DM startup script (STARTUP\_DM). Specify the IDF command in one of the following ways:

- Move the cursor to the new default icon position. Issue the IDF command.

This operation sets the first icon position; the offset of the next icon will be 0,0 (pixels) and the shift vector will be 0,0 (pixels). Therefore, all subsequent icons will appear on top of one another at the first icon position.

- Move the cursor to the new default icon position. Use the <MARK> key or issue the DR command to mark the cursor position. Move the cursor to indicate the offset for the next icon. Issue the IDF command.

This operation sets the first icon position and next icon offset. The shift vector is 0,0 (pixels). Therefore, when icons need to use already-occupied positions, the DM places new icons directly on top of pre-existing icons.

- Move the cursor to the new default icon position. Use the <MARK> key or issue the DR command to mark the cursor position. Move the cursor to indicate the offset for the next icon and once again issue a <MARK> or DR command. Then move the cursor again in order to set the shift vector for reused icon positions.

This operation sets the first icon position and the next icon offset. It also establishes a shift vector so that icons will not appear directly on top of one another if the DM needs to place new icons over pre-existing ones.

- Specify the icon position, offset, and shift explicitly in a command line. The format is as follows:

```
(first_xy_pos)DR:(next_xy_pos)DR:(shift_xy_pos);IDF
```

For example, the command line:

```
Command: (800,10) DR; (850,60) DR; (820,10); IDF
```

places the upper-left corner of the first icon at bit position (800,10), sets the

## IDF (ICON\_DEFAULT)

icon offset vector to (50,50) (found by subtracting the 'initial' from the 'next' bit positions), and sets the shift vector to (20,0) (found by subtracting the 'initial' from the 'shift' bit positions). The next icon would, therefore, appear at bit positions (850,60), the next at (900,110), etc. If an icon must be placed on top of the first icon, then it will be positioned at (820,10), the next at (870,60), etc.

The IDF command requires no arguments or options.

INV (INVERT \_COLOR) -- Set window color.

#### FORMAT

INV [-ON | -OFF]

The INV command sets the color of all windows on monochrome displays. Note that these are *window* backgrounds only; the display background is controlled with BGC (BACKGROUND \_COLOR).

The window color is ON, by default, at login.

NOTE: INV has meaning only for monochromatic displays. It has no effect on nodes with color displays. See the DM command MONO for information about window color on color displays.

#### OPTIONS

If no option is specified, INV toggles the current mode.

- |             |   |
|-------------|---|
| <b>-ON</b>  | Display black characters on a white or green background, depending on display type. |
| <b>-OFF</b> | Display white or green characters on a black background, depending on display type. |

## KBD (KEYBOARD)

**KBD (KEYBOARD) -- Declare keyboard type.**

### FORMAT

#### KBD id

**NOTE:** This command is valid only in the DM file 'NODE\_DATA/STARTUP'; it may not be typed from the keyboard. See the *DOMAIN System User's Guide* for information on startup files.

KBD allows you to specify the keyboard that is attached to your node so that the proper set of standard key definitions may be applied. When this command is invoked in the 'NODE\_DATA/STARTUP file, it causes the DM to execute the corresponding key definition file (/SYS/DM/STD\_KEYS, /SYS/DM/STD\_KEYS2, or /SYS/DM/STD\_KEYS3).

If this command is not invoked in the 'NODE\_DATA/STARTUP file, the DM will do the following. It will first test for a low-profile Model II keyboard and, if one is attached, it will use 'KBD 3'. If a Model II keyboard is not present, the DM will then default to 'KBD 2' (low-profile Model I keyboard).

### ARGUMENTS

id

**(required)**

Specify keyboard ID. Valid IDs are ' ' for the 880 keyboard, '2' for the low-profile Model I keyboard, and '3' for the low-profile Model II keyboard.

**KD (KEY\_DEFINITION) -- Set or display key definition.**

## FORMAT

**KD key\_name [[definition] KE]**

The KD command defines a keyboard key as a sequence of DM commands. It also can display the definition of a key.

## ARGUMENTS

**key\_name**  
(required)

Specify the name of the key to be defined or displayed. Key names are available from **HELP DM KEYS**. Enclose normal alphanumeric and punctuation keys in quotation marks.

**definition**  
(optional)

Specify sequence of DM commands that represent desired key function; separate commands with **NEWLINEs** or semicolons. Definition can be any number of commands, but cannot exceed 256 characters. Definitions may contain other predefined keys (i.e., key definitions may be embedded in one another).

The input request character, '&', which is frequently used in key definitions, must be preceded by an escape character (@) when the KD command appears in a script.

If 'definition' is not specified and KE is present (i.e., definition is null), then the current key definition is deleted and the key reverts to its normal graphic value, if any. If KE is also absent, then the definition of the named key is displayed in the DM message window.

Default if omitted: see above

**KE**  
(optional)

Signal the end of the KD command. This argument is required if 'definition' is present, or if you wish to delete a definition by specifying a null definition.

Default if omitted: display 'key\_name' definition

## EXAMPLES

1. Command: KD L3                      Display definition of key L3.
2. Command: KD F6 AU;TR KE          Define F6 key to move the cursor to end of previous line in window.
3. Command: KD ^C KE                 Delete current definition of CTRL/C.

You can embed key definitions in key definitions, and thereby define keys that define other

## KD (KEY\_DEFINITION)

keys. The embedded key definition follows the same rules as any other key definition. The KE that ends the embedded definition must be separated from the next command by an "escaped" semicolon; that is, a semicolon preceded by the @ character. For example:

Command: KD F3 KD ^X ES 'April is the cruelest month' KE@; PV KE

changes the definition of the F3 key, which normally just invokes the DM command PV, so that it also changes the definition of CTRL/X to print out the string shown. If the ';' were not preceded by an escape character, the definition would not be accepted.

Note that key definitions within key definitions are scanned THREE times: 1) when the outer key definition is made, 2) when the outer key definition is executed and the inner key definition is made, and 3) when the inner key definition is executed. Because of this, you must exercise great care when escaping (with "@") certain special characters such as "@" itself.

**L (LOGIN) -- Log in to a node.**

## FORMAT

**L id [proj [org]] [options]**

The L command allows you to log in to a node. It is valid only at the beginning of a session in response to the "Please log in:" prompt. Typing the L command after logging in causes an error. After entering the L command, the system will request a password. If you specify either the ID or the password incorrectly, the system displays an error message and the correct format of the L command, and you may try again.

If you forget your password, you will have to contact your System Administrator, who can assign a new password to you. The administrator will NOT be able to tell you your current password, since those are encrypted within the system and are not human-readable.

The 'L' character itself is optional when preceded by the "Please log in:" prompt. You may omit it and simply type your ID if you desire.

When you have logged in successfully, the system sets the working directory to your "login home directory," which may be anywhere in your file hierarchy that you please. The login home directory name is first established by the System Administrator for your network when your account is created. You may change it using the -H option (below).

## ARGUMENTS

The 'id', 'proj', and 'org' arguments may be separated either by blanks (as shown), or by periods.

**id**

**(required)**

Specify the user ID assigned to you by the System Administrator when your access privileges were established.

**proj**

**(optional)**

Specify project ID associated with this user id. User IDs may or may not have project IDs, depending on how the access privileges were set up.

**org**

**(optional)**

Specify organization ID associated with this user and project ID. Again, this may or may not be necessary for any particular user ID.

## OPTIONS

**-P**

Set new password. After you have successfully logged on with the old password, the system will prompt you to set a new password. The password may be any combination of ASCII characters up to a maximum of 8 characters. (Note: Some network configurations may require the password to be at least 6

## L (LOGIN)

characters long for added security. Check with your local System Administrator to see if this requirement has been implemented on your network.)

### **-H**

Set new home directory. After you have logged on successfully, the system will prompt you to establish a new home directory for login. Enter the desired pathname. This is the directory which you enter by default each time you log on.

**LCM (LOAD \_COLOR \_MAP) -- Load a color map.**

## FORMAT

**LCM [-p pathname]**

LCM loads a color map from a file which specifies a set of color map entries. Each entry establishes an association between an index and a color value. When the DM is initially loaded, it sets the node's color map from the file in `/sys/dm/color_map`.

If no pathname is given, LCM loads the color map from `/sys/dm/color_map`. In this case, all 16 colors (that is, color entries for color slots 0-15) are reloaded. If you specify a pathname, then LCM reads the given file and tries to load the colors associated with the indexes.

NOTE: IF there are direct mode graphics programs running that have changed the color values for color slots 0-15, then the execution of this command will change the colors in these windows as well as resetting the DM's colors.

## OPTIONS

**-Ppathname** Specify file which contains the color values for red, green, and blue. The format of this file should be identical to the DM's color map file, `/sys/dm/color_map`. For more information about the format of this file, please refer to the manual "Programmer's Guide to DOMAIN Graphics Primitives".

## EXAMPLES

1. Load the DM's color map found in the file `/sys/dm/color_map`.

Command: `lcm`

2. Load the color map specified in the file `my_colormap`.

Command: `lcm -p my_colormap`



**LO (LOG\_OFF) -- Log off a node.**

## FORMAT

### LO [options]

The LO command stops all user processes (except those created by the CPS command and those created by the node startup file 'NODE\_DATA/STARTUP'), closes all pads and files, and returns the display to the "Please log in:" prompt.

If LO cannot terminate all active processes normally, then the command asks you whether or not you wish to blast the remaining processes (see -F below). Respond either "y" or "n".

You may also disable the ability to log off. See -OFF below.

It is possible to execute a DM command script automatically at logout. The logout script must be in a file named 'NODE\_DATA/STARTUP\_LOGOUT.type, where 'type' is one of the standard display type extensions used for startup file names ('.19L', '.COLOR' or none). Note that you cannot start up new processes with CP, CPS, or CPO from this script because the DM is in the process of shutting down all existing processes.

## OPTIONS

- F** Force log off by blasting processes that cannot be stopped normally. If you use this option, be aware that some disk space may be lost. To recover the disk space, use the salvager SALVOL. Files and programs that you had been working with may also be lost. The -F option should only be used as a last resort when the normal logoff procedure is not working.
- OFF** Disable the ability to log off. When this option is specified, the user who is currently logged in will not be able to log off.
- ON** Enable log off. Use this option to restore the ability for a user to log off.

MONO (MONOCHROME)

MONO (MONOCHROME) -- Set color monitor to black and white.

## FORMAT

MONO [-ON|-OFF]

MONO controls whether the DM displays text and windows in color or in black and white. This command operates on color displays only. For information on controlling window color on monochrome displays, see the DM commands BGC (BACKGROUND\_COLOR) and INV (INVERT\_COLOR).

MONO is OFF, by default, at login.

## OPTIONS

If no option is specified, MONO toggles the current mode.

- ON                      Enable monochrome mode.
- OFF                     Disable monochrome mode.

**MSG (MESSAGE) -- Display a message in the DM output window.**

**FORMAT**

**MSG 'string'**

The MSG command instructs the Display Manager to print a string in the DM output window. The string must be enclosed in single quotes.

**ARGUMENTS**

**string**  
**(required)**

Specify the string to be printed in the DM output window.

**EXAMPLES**

The DM command line:

Command: MSG 'Please select another key'

causes the DM to display the message "Please select another key" in the DM output window.

PB (PAD\_BOTTOM)

PB (PAD\_BOTTOM) -- Move bottom of pad into window.

#### FORMAT

##### PB

The PB command moves the bottom line of the pad to the bottom of the current window. This is a pad movement command, as opposed to TB, which moves the cursor to the last line in the window, regardless of that line's position in the pad.

PB does not require either arguments or options.

**PH (PAD\_HORIZONTAL) -- Move pad horizontally by characters.**

**FORMAT**

**PH [-]n**

The PH command moves (scrolls) the pad horizontally under a window in units of characters.

By default, the boxed horizontal arrow keys scroll a pad in 10-character increments.

**ARGUMENTS**

**[-]n  
(required)**

Specify scrolling increment in characters. Positive (unsigned) 'n' scrolls pad left; negative 'n' scrolls pad right.

PN (PAD\_NAME)

PN (PAD\_NAME) -- Save transcript pad in named file.

## FORMAT

### PN pathname

The PN command names a transcript pad and makes it permanent. That is, the pad is stored in a file and remains on the system after all windows to it are deleted. The file remains in use and locked, however, until the process is stopped and all windows are closed. If you do not use the PN command, transcript pads are deleted when all windows to them are deleted.

The PN command can also be used to change the name of an edit pad.

## ARGUMENTS

**pathname**  
**(required)**

Specify the pathname where the DM saves the pad. The pathname must be cataloged in a directory on your node (i.e., you cannot save a file on your node if the file name is cataloged on some other node).

**PP (PAD\_PAGE) -- Scroll pad vertically by pages.**

## FORMAT

**PP [-]n**

The PP command scrolls the pad vertically under a window in units of pages. By default, the boxed up and down arrow keys invoke this command, scrolling in half-page units.

## ARGUMENTS

**[-]n**  
**(required)**

Specify scrolling increment in pages. Positive (unsigned) 'n' scrolls down; negative 'n' scrolls up. Note that 'n' may also be a decimal fraction.

A "page" is defined as the smaller of either of the following values:

- the number of lines that fit in the window
- the number of lines between the bottom of the window and the next form feed or frame

PT (PAD\_TOP)

**PT (PAD\_TOP) -- Move top of pad into window.**

## FORMAT

### PT

The PT command moves the top line of the pad to the top of the current window. This is a pad movement command, as opposed to TT, which moves the cursor to the first line in the window, regardless of that line's position in the pad.

PT does not require either arguments or options.

**PV (PAD\_VERTICAL) -- Scroll pad vertically by lines.**

**FORMAT**

**PV [-]n**

The PV command scrolls the pad vertically under a window in units of lines. By default, the shifted up and down arrow keys on the low-profile keyboard and the F2 and F3 function keys on the 880 keyboard invoke this command, scrolling in one line units.

**ARGUMENTS**

**[-]n**

**(required)**

Specify scrolling increment in lines. Positive (unsigned) 'n' scrolls down; negative 'n' scrolls up.

PW (PAD\_WRITE)

**PW (PAD\_WRITE) -- Update edit file while maintaining edit pad unchanged.**

## FORMAT

### PW

The PW command updates a file that is being edited. It is valid only for writable edit pads. The first time you issue PW, the DM writes the contents of the edit pad to the file that is being edited, without closing the edit pad. The previous contents of the file are saved in a file with the same name and the added suffix .BAK. Subsequent PW or WC (WINDOW\_CLOSE) commands rewrite the new file and leave the .BAK version of the file unchanged.

PW is similar to WC with two exceptions. First, PW leaves the edit pad open. Second, PW writes the new version of the file even if other windows are viewing the edit pad.

PW is useful if, for example, you want to try compiling a program you are editing. If you decide to make additional changes to the program, you can just go back to the edit pad and continue editing, since updates made by PW leave the edit pad open and active.

The <SAVE> key on the low-profile keyboard executes PW;RO to save the pad and put it in read-only mode. There is no predefined key on the 880 keyboard that provides a similar function, although PW is executed along with other DM commands by the default CTRL/Y sequence.

The PW command requires no arguments or options.

**RM (REPLACE\_MARK) -- Replace a mark on the mark stack.**

**FORMAT**

**RM**

The RM command places the last issued mark (DR) back on the mark stack, allowing you to use the mark again.

RM requires no arguments or options.

RO (READ\_ONLY)

RO (READ\_ONLY) -- Set read/write mode.

#### FORMAT

RO [-ON | -OFF]

The RO command puts a pad into (-ON) and out of (-OFF) read-only mode. If no option is supplied, the current mode is toggled. The pad must be in write mode (-OFF) in order for you to insert or delete anything.

By default, CTRL/M invokes the RO command without options to toggle the current mode.

An "R" appears in the window legend of a pad in read-only mode. The "R" disappears in write mode.

An edit pad which has been modified cannot be made read-only without first writing it out with the PW command. See the PW command description for details.

**RS (REFRESH\_SCREEN) -- Refresh screen.**

**FORMAT**

**RS**

The RS command refreshes the entire screen, updating all windows with any pending changes. By default, the CTRL/F sequence invokes this command.

RW (REFRESH\_WINDOW)

RW (REFRESH\_WINDOW) -- Refresh a window.

#### FORMAT

RW [-R]

The RW command causes the DM to refresh the contents of the current window immediately, updating it with any pending changes.

When an unexpected system fault, such as a network failure, occurs, pads may be marked undisplayable in order to avoid further faults. When this happens, the DM displays an error message instead of the window's normal contents. When the problem has been resolved, use the -R option (reset) to redisplay the window's normal contents.

**S (SUBSTITUTE) -- Substitute all occurrences of matched string in defined range.**

## FORMAT

**[range]S[[/[string1]]/string2/]**

The S command substitutes one literal string for a string described by a regular expression over a defined text range. The command does not move the cursor or the window, but does update the window when the substitution is completed. Strings used with this command are also saved for later use (see below).

All substitutions are case sensitive, unlike searches, which ignore case unless told otherwise. Substitution case sensitivity cannot be disabled.

If the Display Manager scans more than 100 lines while processing a substitute command, it displays a "Substitute in progress..." message in its message window. Then it polls for keystrokes for every 10 lines it processes. At this point you may:

1. Wait for DM to complete the operation.
2. Use the keyboard. It works as it does normally. You can type into any pad except the one being searched. You can issue any Display Manager command except another search or substitute command. The Display Manager executes these commands when it completes the substitution. You can type input to another Shell or program (if it was previously waiting for input). Subsequent user process input and output requests will be processed when the Display Manager finishes the substitution.

## ARGUMENTS

If no arguments are specified, the previous substitution will be repeated from the current cursor position to the end of the line.

**range**  
(optional)

Specify range of text in which substitution is to be made. See the section on defining text ranges in the previous chapter for details.

Default if omitted: use current cursor position to end of line

**string1**  
(optional)

Specify string to be replaced in the form of a regular expression. If this argument is omitted but the opening delimiter (/) is used (i.e., "S//string2/"), then string1 defaults to the string1 used in the last *search* operation. If the delimiter is also omitted (i.e., "S/string2/"), then string1 defaults to the string used in the last *substitution* operation.

Default if omitted: see above

**string2**  
(optional)

Specify literal replacement string. This is *not* a regular

## S (SUBSTITUTE)

expression). An "&" can be used to denote string1. If 'string1' is present, then 'string2' is required.

Default if omitted: repeat last substitution command

### EXAMPLES

1. CTRL/T  
<MARK>  
CTRL/B  
<CMD> s/Fielding/Tom Jones/

Move to first character in the pad.  
Place a mark.  
Move to last character in the pad.  
Replace the string "Fielding" with  
"Tom Jones" throughout the marked  
range (in this case, the entire pad).
2. <CMD> s/Tom/& Jones/

Replace "Tom" with "Tom Jones".  
Since no range was marked or specified,  
the replacement takes effect from  
the current cursor position to the  
end of the line.

See the section on "Using Regular Expressions" in the previous chapter for more examples.

SC (SET\_CASE) -- Set search case sensitivity.

#### FORMAT

SC [-ON | -OFF]

A search can be either case-sensitive or case-insensitive. In case-sensitive searches, the characters must match in case (i.e., /mary/ would NOT locate the string "MARY"). In case-insensitive searches, uppercase and lowercase letters are considered equivalent. By default, searches are case-insensitive.

The -ON option explicitly specifies a case-sensitive search; the -OFF switch explicitly specifies a case-insensitive search. Typing the SC command without options toggles the current case comparison setting.

NOTE: The SC command has no effect on *substitution* operations, only *search* operations. Substitutions are always sensitive to the case of the strings involved.

## SHUT

**SHUT -- Shut down system.**

## FORMAT

### SHUT [-F]

The SHUT command exits from the Display Manager and shuts down the system. The Display Manager first closes all windows and pads, then unloads the operating system (AEGIS) and enters the Mnemonic Debugger that resides in the node's boot PROM. If user processes are still active, the SHUT command attempts to stop them. If they stop normally, the shutdown proceeds. If the Display Manager cannot stop them normally, the SHUT command aborts.

The SHUT command has the same effect on system software as turning the node's power off.

To restart the system, type **EX AEGIS** in the Mnemonic Debugger.

To force either log off or shut down, specify the **-F** option. (The same effect can be achieved by replying "y" to a request to blast processes that cannot be closed normally.) If you use this, however, be aware that some disk space may be lost if processes cannot be terminated normally. To recover the disk space, use the salvager **SALVOL**. See the **SALVOL (SALVAGE\_VOLUME)** command.

**SO (SUBSTITUTE\_ONCE) -- Substitute first occurrence of matched string.**

**FORMAT**

**[range]SO[[/[string1]]/string2]**

The SO command is identical to the S (SUBSTITUTE) command except that 'string2' replaces only the first occurrence of 'string1' in each line of the defined range of text.

SQ (SEARCH\_QUIT)

SQ (SEARCH\_QUIT) -- Abort a search operation.

## FORMAT

### SQ

The SQ command aborts a text search, and cancels any action involving the ECHO command. This command is equivalent to ABRT.

The SQ command aborts the current search. The DM returns the message "Search aborted." It does not move the window. Note that you cannot type this command during a search. You must invoke it with a defined key.

When you use SQ to abort the ECHO command, SQ cancels a 'move window with rubberbanding' or 'grow window with rubberbanding' operation; or, it cancels highlighting for a defined range of text, depending on how ECHO was used.

**TB (TO\_BOTTOM) -- Move cursor to bottom line in window.**

**FORMAT**

**TB**

The TB command moves the cursor to the bottom line in the window. This is in contrast to the PB command, which moves the bottom line of the pad into the window.

TB requires no arguments or options.

**NOTE:**

There is a homonymous Shell command: TB (TRACEBACK) -- Print traceback after a fault. See the TB command description in the Shell chapter for details.

TDM (TO\_DM\_WINDOW)

TDM (TO\_DM\_WINDOW) -- Move cursor to DM input window.

## FORMAT

### TDM

TDM moves the cursor to the Display Manager input window (labeled "Command: " at the bottom of the screen) so that you can enter DM commands.

By default, the <CMD> key (L5) invokes the TDM command.

TDM requires no arguments or options.

**TH (TAB\_HORIZONTAL) -- Move cursor right to next tab stop.**

**FORMAT**

**TH**

The TH command moves the cursor right to the next horizontal tab stop. Tabs are global (i.e., they apply to all windows), and may be set with the DM command TS. Initially, tabs are set every 5 spaces.

By default, the <TAB> key invokes the TH command. Note that this does NOT insert an ASCII tab character into the file; it simply positions the cursor to the next tab stop.

TH requires no arguments or options.

THL (TAB\_HORIZONTAL\_LEFT)

THL (TAB\_HORIZONTAL\_LEFT) -- Move cursor left to previous tab stop.

**FORMAT**

**THL**

The THL command moves the cursor left to the next horizontal tab stop. Tabs are global (i.e., they apply to all windows), and may be set with the DM command TS. Initially, tabs are set every 5 spaces.

By default, the CTRL/<TAB> sequence invokes the THL command. Note that this does NOT insert an ASCII tab character into the file; it simply positions the cursor to the previous tab stop.

THL requires no arguments or options.

TI (TO\_INPUT\_WINDOW) -- Move cursor to next input window.

**FORMAT**

**TI**

TI moves the cursor to the next fully unobscured window in which input is accepted (i.e., the next window that opens into neither a transcript nor a read-only edit pad). The cursor is placed at its last previous position in the window.

The Display Manager scans across the screen from left to right and top to bottom to find the next window.

**TL (TO\_LEFT)**

**TL (TO\_LEFT) -- Move cursor to the beginning of the current line.**

**FORMAT**

**TL**

The TL command moves the cursor left to the beginning of the current line. By default, the bar-left arrow key invokes the TL command.

TL requires no arguments or options.

**TLW (TO\_LAST\_WINDOW) -- Move cursor to last (previous) window.**

**FORMAT**

**TLW**

TLW moves the cursor back to the window it was in before it moved to the current window. The cursor is placed at its last previous position in the window.

By default, the CTRL/L sequence invokes the TLW command.

TN (TO\_NEXT\_WINDOW)

TN (TO\_NEXT\_WINDOW) -- Move cursor to next window.

## FORMAT

### TN

TN moves the cursor to the next fully unobscured window on the screen. Any window that is partially covered by another is not considered in the search. The DM scans the screen from top to bottom to find the next window, selecting the one whose upper-left corner is the "highest" (i.e., has the lowest Y coordinate value), then proceeding downward across the screen. If there are panes within a window, the cursor is positioned in the next lower pane until the pane choices have been exhausted, before moving to the next "lower" window. Once the next window is located, the DM places the cursor at its last previous position within that window.

By default, the <NEXT WNDW> key (LB) invokes this command.

**TNI (TO\_NEXT\_ICON) -- Move cursor to next icon.**

**FORMAT**

**TNI**

TNI moves the cursor to the next fully unobscured icon on the screen. Any icon that is partially covered by another is not considered in the search. The DM scans the screen from top to bottom to find the next icon, selecting the one whose upper-left corner is the "highest" (i.e., has the lowest Y coordinate value), then proceeding downward across the screen.

This command is similar to the TN command, which positions the cursor to the next fully unobscured window on the screen.

TR (TO\_RIGHT)

TR (TO\_RIGHT) -- Move cursor to the end of the current line.

## FORMAT

### TR

The TR command moves the cursor right to the end of the current line. By default, the bar-right arrow key invokes the TR command.

TR requires no arguments or options.

**TS (TAB\_SET) -- Set tab stops for all windows.**

## FORMAT

**TS [n1] [n2] ... [-R]**

The TS command sets the default tab stops for all windows. Tab stops may also be set from within a program using a call to the system routine PAD\_\$SET\_TABS; those set under program control override the tab stops set by TS within windows belonging to the program.

By default, tabs are initially set every 5 spaces.

**NOTE:** The DM command "=" displays the line and column numbers of the current cursor position. This can be helpful when trying to set tab stops visually.

## ARGUMENTS

If no arguments are specified, a stop is set at every character on the line.

**n1 n2 ...**

**(optional)**

Specify tab stops. The 'n' values are integers representing absolute character positions. They must appear in increasing order. Columns are numbered starting with one.

Default if omitted: see above

## OPTIONS

**-R**

Repeat the last interval.

## EXAMPLES

Command: TS 7 12 -r

(Set tabs at columns 7 and 12, and every 5 spaces thereafter.)

TT (TO\_TOP)

TT (TO\_TOP) -- Move cursor to top line in window.

## FORMAT

### TT

The TT command moves the cursor to the top line in the window. This is in contrast to the PT command, which moves the top line of the pad to the the top of the window.

TT requires no arguments or options.

**TWB (TO\_WINDOW\_BORDER) -- Move cursor to a specified window border.**

**FORMAT**

**TWB {-L | -R | -T | -B}**

The TWB command moves the cursor to a border of the current window, as specified by the command options. You must specify an option with TWB.

**OPTIONS**

One of the following options is required.

- L**                      Move the cursor to the left window border parallel to the previous cursor position.
- R**                      Move the cursor to the right window border parallel to the previous cursor position.
- T**                      Move the cursor to the top window border directly above the previous cursor position.
- B**                      Move the cursor to the bottom window border directly below the previous cursor position.

## UNDO

**UNDO -- Undo previous DM command(s).**

### FORMAT

#### UNDO

UNDO works by compiling a history of DM activities in input and edit pads in reverse chronological order. Invoking UNDO reverses the effect of the most recent DM command. Successive UNDOs will undo further back in history. Note that this only applies to DM operations; Shell operations (such as compiling a program) cannot be undone.

The UNDO buffers (one per edit pad and one per input pad) are circular lists that, when full, eliminate the oldest entries to make room for new ones. Entries are grouped together in sets. For example, a S (SUBSTITUTE) command may change five lines. While UNDO considers this to be five entries, the five entries are grouped into a single set so that one UNDO will change all five lines back to their original state. When a buffer becomes full, the oldest *set* of entries is erased. This means that UNDO will never partially undo an operation: it will either completely undo it or do nothing.

An edit undo buffer can hold up to 1024 entries. An input undo buffer can hold up to 128 entries.

By default, the <UNDO> key on low-profile keyboards invokes the UNDO command. There is no predefined key for this function on 880 keyboards.

UNDO requires no arguments or options.

WA (WINDOW\_AUTOHOLD) -- Set window autohold mode.

#### FORMAT

WA [-ON | -OFF]

NOTE: Autohold mode applies only to windows open into transcript pads.

The WA command switches a window into (-ON) and out of (-OFF) autohold mode. WA without options toggles the current setting. In autohold mode, the window automatically enters hold mode (in which the contents of a window are temporarily frozen) if either of the following conditions is true:

- A full window of output is available and none of it has been displayed.
- A form feed or create frame operation is output to the pad. In this case, the window displays the output preceding the form feed. When the window exits from hold mode, the output following the form feed or create frame operation starts at the top of the window.

Initially, windows are not in autohold mode. The window legend contains an "A" when the window is in autohold mode.

WC (WINDOW\_CLOSE)

WC (WINDOW\_CLOSE) -- Close window and associated functions.

## FORMAT

WC [entry\_name] [-Q | -F | -A | -S]

The WC command closes (deletes) a window or window group. It may also close the pad into which the window looks, depending on the following conditions.

If other windows into the pad besides the one being closed exist, the DM naturally leaves the pad open. If there are no other windows into the pad, however, the DM closes it. The closed pad is then either deleted (if it was temporary) or saved under its pathname (if it was named and permanent, i.e., a permanent disk file).

If the pad is a writable edit pad, and is being viewed only through the current window, the DM renames the old file by appending .BAK to its name, and writes the edited version to the original file name. If multiple windows are viewing the edit pad, WC simply closes the window -- it does not write the file or rename the old file. To force the DM to write the file and create the .BAK version, use the DM command PW (PAD\_WRITE) or the <EXIT> or CTRL/Y keys (see below).

A transcript (output) window normally cannot be closed if it is the last window into an active process (see -F below).

Note that the DM cannot delete a permanent pad (file); you must use the Shell command DLF (DELETE\_FILE) for this purpose.

Two keys (or control/key sequences) have been predefined to perform related functions:

		low-profile	880		
PW;WC -Q	(or)	<EXIT> (R5)	CTRL/Y		Close window, pad; update file
WC -Q	(or)	<ABORT> (R5S)	CTRL/N		Close window, pad; ignore changes

## ARGUMENTS

**entry\_name**  
(optional)

Specify the name of the window or window group to be closed. If the name of the window or group appears as a text string somewhere on the display, you may use the following time-saving feature: place the cursor on the name, then press <MARK>. Now issue the WC command. WC uses the MARKed name for the 'entry\_name' argument.

Default if omitted: close the window under the cursor.

**OPTIONS**

If no options are specified, WC closes the window and pad, then deletes the pad (if temporary) or rewrites it (if permanent) as described above. Only one of the following options may be specified at a time.

Default options are indicated by "(D)."

- Q                   Quit without updating pad (file). Any changes made while window was open are ignored. The system prompts you with "File modified. OK to quit?" if you have made changes to verify that you really wish to discard them.
- F                   Force window closure, even if this window was the last one open into a process. Note that the process will become inaccessible, however, if no windows are left.
- A                   Enable auto-close for current window. When auto-close is enabled, the current window will close when the pad into which it looks is closed.
- S                   (D)   Disable auto-close for current window. If auto-close is disabled (the default condition), then the current window persists after the pad into which it looks is closed.

WDF (WINDOW\_DEFAULT)

WDF (WINDOW\_DEFAULT) -- Define DM default window positions.

#### FORMAT

[region]WDF [n]

The WDF command lets you define any of the DM's five default window positions. To define a default window position, mark (with the DM command "DR") the region that will display the window, and issue the WDF command.

#### ARGUMENTS

**region**

(optional)

Specify the area of the screen where the new window will be displayed. For details on window regions, see "Defining Points and Regions" elsewhere in this chapter.

Default if omitted: use marked region

**n**

(optional)

Specify the ID number (1-5) of the DM default window that is being defined. If you omit n, the WDF command discards any saved window parameters, so the next window created uses the stock default window boundaries.

Default if omitted: see above

**WG (WINDOW\_GROW) -- Grow or shrink a window.**

## **FORMAT**

**[region]WG**

**NOTE:** There is a companion grow command **WGE** that provides visible feedback during a grow operation. See the **WGE** command description elsewhere in this chapter for information on that command.

The **WG** command changes the size of a window by moving one edge or corner across the screen while leaving the other edges and/or corners where they are. To grow or shrink a window, first mark the edge or corner you want to move by positioning the cursor at that edge or corner and issuing the **DR** command string or pressing **<MARK>**. Then move the cursor to the edge or corner's new location and issue the **WG** command. (By default, **CTRL/G** invokes the **WG** command on low-profile keyboards. This function is not available by default on 880 keyboards.) The marked edge or corner moves to the new cursor position, and the window shrinks or grows accordingly. If you want to move only an edge, move the cursor only in the direction perpendicular to that edge. Moving the cursor in two dimensions causes a corner to move.

## **ARGUMENTS**

**region**  
**(optional)**

Specify old and new locations of edge or corner. May be specified in a variety of formats: see "Defining Points and Regions" in the previous chapter. This argument is **REQUIRED** if you do not use the cursor placement and **<MARK>** operation described above.

Default if omitted: use marked region

WGE (WINDOW\_GROW\_ECHO)

WGE (WINDOW\_GROW\_ECHO) -- Grow/shrink a window with rubberbanding.

## FORMAT

### WGE

The WGE command changes the size of a window. To enlarge or shrink a window with WGE, position the cursor in the window and issue the WGE command. (By default, the <GROW> key invokes the WGE command on the low-profile keyboard, while CTRL/G provides the same function on the 880 keyboard.) After you enter the WGE command, an outline, or "rubberband" will appear to show you the size and shape that the window will take when you complete the grow operation. Move the cursor until the rubberband matches the new size you want for the window. Then issue the DR; ECHO command sequence or press <MARK> to complete the grow operation. You can use the SQ command (CTRL/X) to abort a grow operation using rubberbanding.

WGE requires no arguments or options.

**WGRA (WINDOW\_GROUP\_ADD) -- Create or add to a window group.**

## FORMAT

**WGRA group\_name [entry\_name]**

The WGRA command creates a new window group with the specified group\_name, or adds a window or group to an existing group. By default, if you do not specify an entry\_name (the name of a window or group) WGRA uses the pathname of the window where the cursor was last positioned.

## ARGUMENTS

**group\_name**  
(required)

Specify the name of the group to be created or enlarged.

**entry\_name**  
(optional)

Specify the name of the window or group to be added to 'group\_name'.

Default if omitted: Use the pathname of the window where the cursor was last positioned.

## EXAMPLES

Command: WGRA Shell\_Windows Process\_1

This command adds a window called "Process\_1" to a group of windows called "Shell\_Windows".

WGRR (WINDOW\_GROUP\_REMOVE)

WGRR (WINDOW\_GROUP\_REMOVE) -- Remove window/group from group.

## FORMAT

**WGRR group\_name [entry\_name]**

The WGRR command removes a window or group from a window group. By default, if you do not specify an entry\_name (the name of a window or group) WGRR uses the pathname of the window where the cursor was last positioned.

## ARGUMENTS

**group\_name**  
(required)

Specify the name of the window group that contains the window or group you want to remove.

**entry\_name**  
(optional)

Specify the name of the window or group to be removed.

Default if omitted: use the pathname of the window where the cursor was last positioned

## EXAMPLES

Command: WGRR Shell\_Windows Process\_2

This command removes a window called "Process\_2" from a group of windows called "Shell\_Windows".

WH (WINDOW\_HOLD) -- Set window hold mode.

#### FORMAT

WH [-ON | -OFF]

NOTE: Hold mode applies only to windows open into transcript pads.

The WH command switches a window into (-ON) or out of (-OFF) hold mode. WH without options toggles the current setting. In hold mode, the contents of the window are "frozen" and do not change when a program sends more output to the pad. When a window is not in hold mode, the window automatically moves to the end of the pad as new output appears.

By default, the <HOLD> key on low-profile keyboards and the <HOLD/GO> key on 880 keyboards invoke the WH command.

Initially, windows are not in hold mode. The window legend contains an "H" when the window is in hold mode.

WI (WINDOW \_INVISIBLE)

WI (WINDOW \_INVISIBLE) -- Make a window or group visible or invisible.

## FORMAT

WI [entry\_name] [-W | -I]

The WI command controls the visibility or invisibility of the specified window or group. WI without options toggles the current mode. By default, if you do not specify an entry\_name (the name of a window or group) WI uses the name of the window under the cursor.

## ARGUMENTS

**entry\_name**  
(optional)

Specify the name of the window or group you want to make visible or invisible. If the name of the window or group appears as a text string somewhere on the display, you may use the following time-saving feature: place the cursor on the name, then press <MARK>. Now issue the WI command. WI uses the MARKed name for the 'entry\_name' argument.

Default if omitted: manipulate the window under the cursor.

## OPTIONS

If no option is specified, WI toggles the current visibility setting.

- I** Force the window or group to be invisible.
- W** Force the window or group to appear as a window.

**WM (WINDOW\_MOVE) -- Move a window across the screen.**

## **FORMAT**

**[region]WM**

**NOTE:** There is a companion move command **WME** that provides visible feedback during a move operation. See the **WME** command description elsewhere in this chapter for information on that command.

The **WM** command moves a window across the screen. The **DM** moves the window whose nearest unobscured edge or corner is the first point of the region. The new location of this edge or corner is the second point of the region. So, to move a window, place the cursor at one corner of the window you want to move and issue the **DR** command (or press **<MARK>**). Next, place the cursor at the desired new position of the corner. Finally, issue the **WM** command.

If you do not define a region, the **WM** command causes the nearest window corner to be moved to the current cursor position. This can cause unexpected results if there are multiple windows on the screen, however, since your idea of the "nearest" window may not be the same as the **DM**'s. In that case, it is safer to **<MARK>** the window you want to move.

## **ARGUMENTS**

**region**

**(optional)**

Specify old and new locations of edge or corner. This may be specified in a variety of formats: see "Defining Points and Regions" in the previous chapter.

Default if omitted: use current cursor position

WME (WINDOW\_MOVE\_ECHO)

WME (WINDOW\_MOVE\_ECHO) -- Move a window using rubberbanding.

## FORMAT

### WME

The WME command moves a window across the screen using the rubberbanding feature. To move a window, place the cursor in the window you want to move and issue the WME command. (By default, the <MOVE> key invokes the WM command on the low-profile keyboard while CTRL/W provides the same function on the 880 keyboard.) After you issue the WME command an outline or "rubberband" will appear to show you where the window will be when you complete the move operation. Now move the cursor until the rubberband is at the desired window position. Finally, issue the DR; ECHO command or press <MARK> to complete the grow operation. You can use the SQ command (CTRL/X) to abort a move operation using rubberbanding.

The WME command requires no arguments or options.

**WP (WINDOW\_POP) -- Push or pop a window on the stack.**

## FORMAT

**WP [entry\_name] [-T|-B]**

The WP command pops a window to the top of the stack or pushes a window to the bottom of the stack. If the cursor rests in a partially obscured window, the WP command pops the window to the top of the pile. If the cursor rests in a completely visible window, WP pushes the window to the bottom of the pile. WP can also manipulate specific named windows or window groups. See the ARGUMENTS section below.

By default, the WP command is invoked by the <POP> key on low-profile keyboards, while the same function is provided by CTRL/P on 880 keyboards.

## ARGUMENTS

**entry\_name**  
(optional)

Specify the name of the window or group you want to push or pop. If the name of the window or group appears as a text string somewhere on the display, you may use the following time-saving feature: place the cursor on the name, then press <MARK>. Now issue the WP command. WP uses the MARKed name for the 'entry\_name' argument.

Default if omitted: push or pop the window under the cursor.

## OPTIONS

The following options are intended primarily for use in DM scripts, where you may not be able to predict the presence of other windows on the screen.

- T** Force a window to the top of the window stack.
- B** Force a window to the bottom of the window stack.

## EXAMPLES

Command: wp -t                      Pop the window containing the cursor to the top of the window stack.

Command: wp slide -b                Push the window named 'slide' to the bottom of the stack.

WS (WINDOW\_SCROLL)

WS (WINDOW\_SCROLL) -- Set window scroll mode.

## FORMAT

WS [-ON | -OFF]

NOTE: Scroll mode applies only to windows open into transcript pads.

The WS command switches a window into (-ON) and out of (-OFF) "line-at-a-time" scrolling. WS without options toggles the current setting. When "line-at-a-time" scrolling is in effect, output appears in the window one line at a time, scrolling past. When "line-at-a-time" scrolling is not in effect, output appears a window at a time.

By default, CTRL/S invokes the WS command.

Initially, all windows (except edit windows) have "line-at-a-time" scrolling. The window legend contains an "S" when the window is in scroll mode.

**XC (COPY) -- Copy text to paste buffer.**

## FORMAT

**[range]XC [-R] [-F pathname | name]**

The XC command copies a range of text from any pad into a paste buffer or system file. The copied text remains undisturbed.

By default, the <COPY> key on low-profile keyboards and CTRL/C on 880 keyboards invoke the XC command using the default (unnamed) paste buffer.

## ARGUMENTS

**range**  
**(required)**

Specify range of text to be copied. Define the range to be copied as described in "Defining a Range of Text" in the previous chapter.

Default if omitted: copy from cursor to end of line

**name**  
**(optional)**

Specify paste buffer name. Text is written to the named buffer. If text is copied to a buffer that has previously been used, the new text overwrites the old. You may have up to 100 buffers open per log in session.

Default if omitted: use unnamed buffer

## OPTIONS

**-F pathname**

Specify system file to receive copied text. If the file already exists, the copied text overwrites the current file contents. Not valid if 'name' argument is present.

**-R**

Specify copy for a rectangular portion of text.

**XD (CUT)**

**XD (CUT) -- Cut (delete) text and write it to paste buffer.**

## FORMAT

**[range]XD [-R] [-F pathname | name]**

The XD command copies a range of text into a paste buffer or system file, then deletes the text from the pad. This command can be used only in a writable pad.

By default, the <CUT> key on low-profile keyboards and CTRL/E on 880 keyboards invoke the XD command using the default (unnamed) paste buffer.

## ARGUMENTS

**range**

**(required)**

Specify range of text to be cut. Define the range to be cut as described in "Defining a Range of Text" in the previous chapter.

Default if omitted: cut from cursor to end of line

**name**

**(optional)**

Specify paste buffer name. Text is written to the named buffer. If text is copied to a buffer that has previously been used, the new text overwrites the old. You may have up to 100 buffers open per log in session.

Default if omitted: use unnamed buffer

## OPTIONS

**-F pathname**

Specify system file to receive cut text. If the file already exists, the cut text overwrites the current file contents. Not valid if 'name' argument is present.

**-R**

Specify cut for a rectangular portion of text.

**XI (COPY\_IMAGE) -- Copy a display image into a graphics map file.**

#### FORMAT

**[range]XI [-F pathname]**

The XI command copies a display image into a graphics map file (GMF). If you do not mark the portion of the display window you want to copy, XI copies the entire window where the cursor is positioned. You can print the GMF using the PRF Shell command with the -PLOT option.

#### ARGUMENTS

**range**

**(optional)**

Specify range of image to be copied. Define the range to be copied as described in "Defining a Range of Text" in the previous chapter.

Default if omitted: copy current window

#### OPTIONS

**-F pathname**

Specify GMF output file. If this option is omitted, the image is written to 'NODE\_DATA/PASTE\_BUFFERS/DEFAULT.GMF'. You can print the GMF using the PRF Shell command with the -PLOT option.

XP (PASTE)

XP (PASTE) -- Paste (write) buffered text into pad.

## FORMAT

XP [-R] [-F pathname | name]

The XP command inserts the contents of a paste buffer or system file into a pad at the current cursor position. The contents of the paste buffer or file are unchanged by this command, making multiple insertions possible. This command can be used only in a writable pad.

By default, the <PASTE> key on low-profile keyboards and CTRL/O on 880 keyboards invoke the XP command using the default (unnamed) paste buffer.

## ARGUMENTS

**name**  
(optional)

Specify paste buffer name. Text is copied from the named buffer. You may have up to 100 buffers open per log in session.

Default if omitted: use unnamed buffer

## OPTIONS

**-F pathname**

Specify system file to provide paste text. Not valid if 'name' argument is present.

**-R**

Specify paste of a rectangular portion of text.

## Chapter 3 Shell Basics

This chapter summarizes the basic concepts that apply to the Shell commands described individually in the following chapter. For a more indepth discussion of these concepts, please refer to the *DOMAIN System User's Guide*.

### 3.1. Command Format

In the most general sense, the operating system has no commands. There are simply files that the Shell looks for and executes. When you type "date" in the Shell input window, the Shell looks for a file called DATE (following its command search rules) and executes it. This means that any files that you create can be given to the Shell for execution. (Of course, if you tell the Shell to execute a file containing nonbinary data -- like the text of a memo -- you will receive an error message.) The point is that any file, no matter where it comes from, may be given to the Shell for interpretation and execution.

The simplest command line consists of a command name followed by arguments to the command, separated by spaces:

```
$ command arg1 arg2 ... argn <RETURN>
```

Normally, you enter one command per line. You may continue a command over several lines by typing @<RETURN> at the end of each line to be continued. The Shell then prompts with "\$\_" to indicate that the current line continues the previous one.

#### 3.1.1. Arguments

The command Shell, which we supply, handles commands that accept multiple arguments (see Figure 3-1). Usually, those arguments come in two forms: a pathname designating a file on which to operate or some other sort of literal string for manipulation, and instructions for special command action. Those arguments that specify special action are almost always optional, and are immediately preceded by a hyphen (-). (The hyphen is necessary because these arguments often require secondary arguments of their own. The commands use the hyphen to interpret correctly where all the different arguments apply.) These special arguments are labeled "Options" in the command descriptions in Chapter 4.

```
$ ld my_dir -len
| | | |
| | | | Option (command modifier)
| | | | Argument (object of command action)
| | | | Command (LIST_DIRECTORY)
| | | | Shell prompt
```

Figure 3-1. Typical Shell Command Format

### 3.1.2. Separators

Normally, Shell commands are separated from each other by carriage returns (NEWLINE characters). You may place multiple commands on the same line by separating the commands with semicolons, up to a total of 256 characters per line. For example,

```
$ wd //mydir/sub1;ld
```

This command line sets your working directory to the directory "//mydir/sub1" and then lists the contents of that directory.

Multiple commands may also appear on a single line when you are using pipes and filters.

### 3.1.3. Node Specifications

Many Shell commands require you to identify a target node on which the commands are to operate. For example, the CRP (CREATE\_A\_PROCESS) command needs to know which node is going to host the new process. You identify nodes with a *node specification*.

A node specification permits a node's communications software to locate other nodes in a local ring or in an internet (a network composed of individual network rings joined via DOMAIN/BRIDGES). This node specification may be either an internet address or a node name.

#### *Internet Addresses*

An internet address has the format:

```
[net].node_id
```

The **net** represents a network number and the **node\_id** represents a hexadecimal node ID. A network number of 0 refers to the local network.

If a node is cataloged (in either your local cache or the NS\_HELPER database), then you can omit the network number when you use an internet address. When you provide only the node ID, the system obtains the network number from either your local cache or the NS\_HELPER database. If you choose to provide a complete internet address, however, the system attempts to locate the node only on the network you specify. Thus, if you specify an incorrect network number, the system will look for the node only on the network that you specify and then report an error; the system will *not* attempt to locate the node on another network.

If a node is *not* cataloged, the system cannot obtain a network number if you omit it. In this case, the system assumes that the node is on the local ring. Thus, for an uncataloged node on the local network, you must provide the node ID, but the network number is optional. However, you must provide both the network number and node ID for an uncataloged node on a remote network.

#### *Node Names*

A node name has the format:

```
//node_name
```

You can use a node name as a node specification only if the node is cataloged (in either your local

cache or the the NS\_HELPER database.) When you use a node name, the system obtains the internet address associated with that name. If a node is not cataloged, you must use an internet address to specify the node.

Note that both disked and diskless nodes can be cataloged and named.

### *Node Specification Examples*

The following examples illustrate ways you can specify a node with an ID of A105, a name of //CASEY, and a network number of 4051237A. (These examples assume that //CASEY is cataloged in the NS\_HELPER database.)

1. \$ LUSR -N 0A105 (Note that hex IDs that start with a letter must be preceded by a '0' for the Shell to parse them correctly.)
2. \$ LUSR -N //CASEY
3. \$ LUSR -N 4051237A.A105

In addition, if you are using a node on ring 4051237A, you can use the following internet address to refer to //CASEY:

```
$ LUSR -N 0.A105 ('0' indicates the local network.)
```

## **3.2. Using Special Characters**

The Shell recognizes a variety of special characters that allow you to change the action of commands. The characters in Table 3-1 have special meanings when they appear on a command line. Note that, while some of these characters have already been discussed as having special meanings in Display Manager commands, regular expressions, and so forth, those meanings do not necessarily carry over to the command Shell environment. Please be careful to keep the different meanings distinct: regular expressions appearing in Shell commands, for instance, should be enclosed in quotation marks to avoid confusion.

The at sign (@) is the Shell's **escape character**. You can place an "@" anywhere on the command line to suppress the special meaning of the next character (including the "@" character itself).

For a full discussion of the usage of Shell special characters, please refer to the *DOMAIN System User's Guide*.

**Table 3-1. Command Shell Special Characters**

<u>Character</u>	<u>Usage</u>
?	Match any single character except NEWLINE.
%	Match zero or more characters up to but not including the period.
*	Match zero or more occurrences of the preceding character.
[string]	Match any single character in the character class "string".
[~string]	Match any character except those in "string".
...	Match zero or more subordinate directories.
=	Copy (derive) leafname from previous argument.
(names)	Group pathnames for use in later derived names.
{expr}	Tag expression for later use.

Figure continued on next page . . . .

### 3.3. The Command Line Parser

Many Shell commands that we supply share a standard command line parsing procedure. It determines how each command processes command line information. Chapter 4 of this manual, and the on-line HELP files, identify commands that use the command line parser. These commands support the following features:

1. You may use wildcards to specify existing pathnames.
2. You may use derived names to specify logically-related pathnames, and parentheses to create several derived names with one command line. (See Table 3-1.)
3. When pertinent, you may include multiple pathnames as command line arguments. For example, "PRF file1 file2 file3".
4. You may use the asterisk character (\*) to cause commands to read pathnames from standard input or from another file. For example,

```
$ prf */fred/names_file
```

prints the files listed in /FRED/NAMES\_FILE. Also,

**Table 3-1. Command Shell Special Characters (cont.)**

Input/Output Control

<u>Character</u>	<u>Usage</u>	<u>Notes</u>
<	Redirect standard input	(3)
<?	Redirect error input	(3)
<</	Read in-line data from standard input	(3)
<<?/	Read in-line data from error input	(3)
>	Redirect standard output	(3)
>?	Redirect error output	(3)
>>	Append standard output	(3)
>>?	Append error output	(3)
	Pipe standard output	(1)
( )	Group commands for I/O redirection	(1)

Parsing Operators

<u>Character</u>	<u>Usage</u>	<u>Notes</u>
#	Comment line in a command file	(4)
&	Run a program or command in the background	(1)
^	Insert parameter	(3)
!	Insert parameter and rescan	(3)
^"cmd"	Insert output of "cmd", with expansion	(3)
^'cmd'	Insert output of "cmd", no expansion	(3)
;	Separate commands on a line	(1)
"	Quoted string, with expansion	(4)
'	Quoted string, no expansion	(4)
@	Escape character	(5)
	Space	(2)

Notes:

- (1) Special anywhere; causes a new command to start.
- (2) Special anywhere; causes a new argument to start.
- (3) Special anywhere; does not start a new argument.
- (4) Special only at the beginning of an argument.
- (5) Special only when immediately preceding a character that would otherwise be special.

```

$ PRF *
  file1
  file2
  file3
  ***EOF***
$

```

reads the names "file1", "file2", and "file3" from standard input, and prints each file. When using the keyboard for standard input, a NEWLINE and an end-of-file character must follow the last name. By default, CTRL/Z generates an end-of-file character.

If you include more than one name on an input line, in standard input or in a names file, the command interprets all names except the first one on each line as derived names. For example,

```

$ CHN *           is equivalent to     $ CHN * =.old
  a a.old         a
  b b.old         b
  c c.old         c
  ***EOF***      **EOF***
$                 $

```

Do not confuse the action of the "\*" character with that of the input redirection symbol "<". The "\*" character causes a Shell command to read *pathnames* from standard input or from another file. The "<" symbol causes a Shell command to read *data* from a file.

### 3.3.1. Standard Command Options

All Shell commands that we supply support the following standard options:

- HELP            Display detailed usage information.
- USAGE          Display brief usage summary.
- VERSION        Display software version number.

**NOTE:**            Using any of these three standard options precludes using any other options within the same command.

### 3.3.2. Command Line Parser Options

Commands that use the command line parser also support the following options (D indicates a default option):

- AE            Abort if a name in pathname cannot be found. If omitted, processing continues to the next name.
- NQ    (D)    Do not issue query to verify wildcard names.
- QW            Issue query to verify wildcard names.

- QA** Issue query to verify all names.
- (hyphen alone) Read further data from standard input. End input with CTRL/Z.
- \*[pathname]** Read file specified for further pathname arguments. If pathname is omitted, read standard input for further pathname arguments.

Commands that delete or modify objects automatically verify names specified with wildcards. You can suppress this query using **-NQ**, or extend it to all names using **-QA**.

When you select a query option, the command writes the selected names to the error output stream with a **?** to prompt you for a response. Then it reads your response from the error input stream (normally the keyboard).

<b>If you respond:</b>	<b>The command:</b>
<b>h</b>	Displays help information.
<b>y</b>	Operates on the name.
<b>n</b>	Ignores the name.
<b>q</b>	Quits immediately.
<b>g</b>	Operates on the name and suppresses further name queries.
<b>d new_default</b>	Resets the default. The Shell performs the default action when it receives a null line query response (i.e., when you simply press <b>&lt;RETURN&gt;</b> ). To change the default, enter <b>d</b> followed by "yes", "no", or "none". The initial default is "none", which means that the command ignores null line responses, and requires explicit yes or no responses.

Chapter 4 describes each Shell command in detail. Those commands that use the command line parser refer you to this section for information on the standard options to avoid repetition in the text.



## Functional Command Index

### Manipulating Files/Directories

#### *Creating Files/Directories*

CRD (CREATE\_DIRECTORY) -- Create a directory  
CRF (CREATE\_FILE) -- Create a file  
CRL (CREATE\_LINK) -- Create a link  
CRRGY (CREATE\_REGISTRY) -- Create or modify network registry

#### *Cataloging Files*

CTOB (CATALOG\_OBJECT) -- Catalog an object  
UCTOB (UNCATALOG\_OBJECT) -- Uncatalog a pathname without deleting the object  
LD (LIST\_DIRECTORY) -- List contents of a directory  
ND (NAMING\_DIRECTORY) -- Set or display naming directory  
WD (WORKING\_DIRECTORY) -- Set or display working directory

#### *Changing File/Directory Attributes*

CHN (CHANGE\_NAME) -- Change an object's name  
CVT\_REC\_UASC -- Convert files between types  
OBTY (OBJECT\_TYPE) -- Set or display the type of an object

#### *Copying Files*

CATF (CATENATE\_FILE) -- Catenate files and write to output  
CPF (COPY\_FILE) -- Copy a file  
CPBOOT (COPY\_BOOT) -- Copy system boot file  
CPSCR (COPY\_SCREEN) -- Copy the display to a file  
CPT (COPY\_TREE) -- Copy a tree  
CRPAD (CREATE\_PAD) -- Create a transcript pad and window  
MVF (MOVE\_FILE) -- Move a file  
TEE -- Copy input to output and to named files

#### *Comparing Files*

CMACT (COMPARE\_ACCOUNT\_FILE) -- Compare account files  
CMPPO (COMPARE\_PPO\_FILE) -- Compare person, project, or organization names  
CMF (COMPARE\_FILE) -- Isolate differences between files  
CMSRF (COMPARE\_SORTED\_FILES) -- Isolate differences between sorted files  
CMT (COMPARE\_TREE) -- Compare all objects in trees

### *Printing Files*

PRF (PRINT\_FILE) -- Print a file on a printer

### *Deleting Files/Directories*

DLF (DELETE\_FILE) -- Delete a file

DLL (DELETE\_LINK) -- Delete a link

DLT (DELETE\_TREE) -- Delete a tree

### *Salvaging Files/Directories*

SALD (SALVAGE\_DIRECTORY) -- Salvage a directory

SALVOL (SALVAGE\_VOLUME) -- Verify and correct allocation of disk blocks

### *Protecting Files/Directories*

ACL (ACCESS\_CONTROL\_LIST) -- List or copy an access control list

EDACL (EDIT\_ACCESS\_CONTROL\_LIST) -- Edit or list an existing ACL

SALACL (SALVAGE\_ACCESS\_CONTROL\_LIST) -- Salvage an ACL

UMASK -- Set DOMAIN/IX file-creation mode mask.

### *Backing Up Files/Directories*

ARCF (ARCHIVE\_FILE) -- Maintain an archive file

RBAK (READ\_BACKUP) -- Restore or index a magnetic tape backup file

WBAK (WRITE\_BACKUP) -- Create a magnetic tape backup file

### *Locking/Unlocking Files*

LKOB (LOCK\_OBJECT) -- Lock an object

LLKOB (LIST\_LOCKED\_OBJECTS) -- List locked objects

ULKOB (UNLOCK\_OBJECT) -- Unlock an object

### *Reading and Writing Files on Tape*

EDMTDESC (EDIT\_MAGTAPE\_DESCRIPTOR) -- Create or modify magtape descriptor file

RWMT (READ\_WRITE\_MAGTAPE) -- Read and write files on magnetic tape

RBAK (READ\_BACKUP) -- Restore or index a magnetic tape backup file

WBAK (WRITE\_BACKUP) -- Create a magnetic tape backup file

### **Editing Files**

#### *Locating Text*

FPAT (FIND\_PATTERN) -- Find a text pattern in a file

FPATB (FIND\_PATTERN\_BLOCK) -- Find blocks of lines containing a pattern

### *Replacing Text*

CHPAT (CHANGE\_PATTERN) -- Replace pattern in text file  
DLDUPL (DELETE\_DUPLICATE\_LINES) -- Strip repeated lines from a file  
ED (EDIT) -- Edit a text file  
EDACCT (EDIT\_ACCOUNT) -- Edit registry account file  
EDPPO (EDIT\_PPO) -- Edit registry PPO files  
EDSTR (EDIT\_STREAM) -- Edit a stream  
MACRO -- Expand macro definitions  
OS (OVERSTRIKE) -- Convert ASCII to FORTRAN carriage control  
TLC (TRANSLITERATE\_CHARACTER) -- Replace characters

### *Sorting Text*

CREFS (CROSS\_REFERENCE\_SYMBOLS) -- Cross-reference symbols in file  
EXFLD (EXTRACT\_FIELDS) -- Manipulate fields of data  
FLEN (FILE\_LENGTH) -- Count lines, words, and characters in a file  
LAMF (LAMINATE\_FILE) -- Laminate files  
REVL (REVERSE\_LINES) -- Reverse each line in a text file  
SRF (SORT\_FILE) -- Sort and/or merge text files

### **Formatting Files**

FMC (FORMAT\_MULTI\_COLUMN) -- Format text file into multiple columns  
FMT (FORMAT\_TEXT) -- Format text file  
OS (OVERSTRIKE) -- Convert ASCII to FORTRAN carriage control  
PAGF (PAGINATE\_FILE) -- Paginate a file to output

### **Developing Programs**

#### *Compiling Programs*

MACRO -- Expand macro definitions

#### *Debugging Programs*

ABTSEV (ABORT\_SEVERITY) -- Set or display abort severity level  
DEBUG -- Invoke the Language Level Debugger  
ESA (EXTERNAL\_SYMBOL\_ADDRESS) -- Display address of external symbol in installed library  
HPC (HISTOGRAM\_PROGRAM\_COUNTER) -- Make a histogram of the program counter  
STCODE (STATUS\_CODE) -- Translate status code value to text message  
TB (TRACEBACK) -- Print traceback after a fault

#### *Loading Programs*

BIND -- Combine object modules into an executable file  
INLIB (INSTALL\_LIBRARY) -- Install a user-supplied library  
LBR (LIBRARIAN) -- Create an object module library

## Managing Your Node

CALENDAR (SET SYSTEM CALENDAR) -- Set system calendar clock  
CPBOOT (COPY\_BOOT) -- Copy system boot file  
FIND\_ORPHANS -- Locate and catalog uncataloged objects  
LVOLFS (LIST\_VOLUME\_FREE\_SPACE) -- List free space on all logical volumes  
SALVOL (SALVAGE\_VOLUME) -- Verify/correct allocation of disk blocks (see Appendix D)  
SCRTO (SCREEN\_TIMEOUT) -- Set/show screen timeout  
SHUTSPM -- Shut down SPM on a node.  
TPM (TOUCH\_PAD\_MODE) -- Set characteristics for the touchpad

## Requesting Process/System Information

### *Process*

CSR (COMMAND\_SEARCH\_RULES) -- List or define command search rules  
DSPST (DISPLAY\_PROCESS\_STATUS) -- Display process status graphically  
FST (FAULT\_STATUS) -- Display fault status information  
LOPSTR (LIST\_OPEN\_STREAMS) -- List open streams  
PST (PROCESS\_STATUS) -- List process internal state information  
STCODE (STATUS\_CODE) -- Translate status code value to text message

### *Node*

BLDT (BUILD\_TIME) -- Display time at which system was built  
DATE -- Display current date and time  
TZ (TIME\_ZONE) -- Set or display system time zone  
LAS (LIST\_ADDRESS\_SPACE) -- List objects mapped into the address space  
LLKOB (LIST\_LOCKED\_OBJECTS) -- List locked objects  
LVOLFS (LIST\_VOLUME\_FREE\_SPACE) -- List free space on all logical volumes  
NETSTAT (NETWORK\_STATISTICS) -- Display network statistics  
NETSVC (NETWORK\_SERVICE) -- Set or display network services  
TCTL (TERMINAL\_CONTROL) -- Set or display terminal (SIO line) characteristics

### *Network*

LCNET (LIST\_CONNECTED\_NETWORKS) -- Display internet routing information  
LCNODE (LIST\_CONNECTED\_NODES) -- List nodes connected to the network  
LUSR (LIST\_USER) -- List users logged on  
NETMAIN (NETWORK\_MAINTENANCE) -- Control/analyze network maintenance statistics  
NETMAIN\_NOTE (NETWORK\_MAINTENANCE\_NOTES) -- Place message in network error log  
NETSTAT (NETWORK\_STATISTICS) -- Display network statistics  
NETSVC (NETWORK\_SERVICE) -- Set or display network services  
PROBENET (PROBE\_NETWORK) -- Probe network and display error statistics  
RTCHK (ROUTING\_CHECK) -- Test traffic between adjacent routers  
RTSTAT (ROUTING\_STATISTICS) -- Display internet router information  
RTSVC (ROUTING\_SERVICE) -- Set or display internet routing service

## Setting Process Conditions

### *Controlling Programs*

ABTSEV (ABORT\_SEVERITY) -- Set or display abort severity level  
FPPMASK (FLOATING\_POINT\_MASK) -- Set or display floating-point error mask  
PPRI (PROCESS\_PRIORITY) -- Set or display process priority  
SIGP (SIGNAL\_PROCESS) -- Signal a process to stop  
SHUTSPM -- Shut down SPM on a node.  
RETURN -- Return from the current Shell level at a specific error value

### *Controlling Shells*

SIGP (SIGNAL\_PROCESS) -- Signal a process to stop  
SHUTSPM -- Shut down SPM on a node.  
CRP (CREATE\_A\_PROCESS) -- Create a process on a remote node  
CRSUBS (CREATE\_SUBSYSTEM) -- Create a protected subsystem  
CSR (COMMAND\_SEARCH\_RULES) -- List or define command search rules  
ENSUBS (ENTER\_SUBSYSTEM) -- Enter a protected subsystem at Shell command level  
RDYM (READY\_MESSAGE) -- Set system ready message  
SUBS (SUBSYSTEM) -- Set or display subsystem attributes  
SH (SHELL) -- Invoke a Shell (command line interpreter)  
SET -- Set current Shell conditions  
BON -- reset SHELL -B flag  
BOFF -- reset SHELL -B flag  
EON -- reset SHELL -E flag  
EOFF -- reset SHELL -E flag  
VOFF (VERIFY\_OFF) -- Reset SHELL -V flag  
VON (VERIFY\_ON) -- Reset SHELL -V flag  
XOFF -- Reset SHELL -X flag  
XON -- Reset SHELL -X flag  
XSUBS (EXECUTE\_SUBSYSTEM) -- Execute a Shell script-protected subsystem manager  
LOGIN -- Log in to a running process

### **Writing Your Own Commands (Shell Scripts)**

ARGS (ARGUMENTS) -- Echo command line arguments  
EQS (EQUALS) -- Compare strings for equality  
EXISTF -- Check to see if an object exists  
EXIT -- Exit a loop  
NEXT -- Return to the top of a loop  
RETURN -- Return from current Shell level  
IF -- Execute a conditional statement  
FOR -- Execute a FOR loop  
SELECT -- Execute a SELECT condition  
WHILE -- Execute a WHILE loop  
XDMC (EXECUTE\_DM\_COMMAND) -- Execute a Display Manager command  
SOURCE -- Execute a Shell script at the current Shell level

## Using Shell Variables

DLVAR (DELETE\_VARIABLE) -- Delete a Shell variable  
EXISTVAR (EXIST\_VARIABLE) -- Check to see if a variable exists  
EXPORT -- Change a Shell variable into an environment variable  
LVAR (LIST\_VARIABLES) -- List name, type, and value of current variables  
READ -- Set variables equal to input values  
READC -- Set variables equal to input character values  
READLN -- Set a variable equal to an input value

## Managing Network Functions

### *Manipulating the Network Registry*

CMACCT (COMPARE\_ACCOUNT\_FILE) -- Compare account files  
CMPPO (COMPARE\_PPO\_FILE) -- Compare person, project, or organization names  
CRRGY (CREATE\_REGISTRY) -- Create or modify network registry  
EDACCT (EDIT\_ACCOUNT) -- Edit registry account file  
EDNS (EDIT\_NAMING\_SERVER\_HELPER) -- Invoke editor for NS\_HELPER  
EDPPO (EDIT\_PPO) -- Edit registry PPO files  
LRGY (LIST\_REGISTRY) -- List contents of registry files  
MRGRGY (MERGE\_REGISTRIES) -- Merge two network registries  
SALRGY (SALVAGE\_REGISTRY) -- Salvage network and local registries

### *Managing an Internet*

LCNET (LIST\_CONNECTED\_NETWORKS) -- Display internet routing information  
LCNODE (LIST\_CONNECTED\_NODES) -- List nodes connected to the network  
LUSR (LIST\_USER) -- List users logged on  
LVOLFS (LIST\_VOLUME\_FREE\_SPACE) -- List free space on all logical volumes  
NETMAIN (NETWORK\_MAINTENANCE) -- Control/analyze network maintenance statistics  
NETMAIN\_NOTE (NETWORK\_MAINTENANCE\_NOTES) -- Place message in network error log  
NETSTAT (NETWORK\_STATISTICS) -- Display network statistics  
NETSVC (NETWORK\_SERVICE) -- Set or display network services  
PROBENET (PROBE\_NETWORK) -- Probe network and display error statistics  
RTCHK (ROUTING\_CHECK) -- Test traffic between adjacent routers  
RTSTAT (ROUTING\_STATISTICS) -- Display internet router information  
RTSVC (ROUTING\_SERVICE) -- Set or display internet routing service

### *Controlling Peripheral Devices*

EDMTDESC (EDIT\_MAGTAPE\_DESCRIPTOR) -- Create or modify magtape descriptor files  
NETSVC (NETWORK\_SERVICE) -- Set or display network services  
PRSVR (PRINT\_SERVER) -- Start the Print Server

### *Controlling Logical Volumes*

CTNODE (CATALOG\_NODE) -- Catalog a node in the network  
UCTNODE (UNCATALOG\_NODE) -- Uncatalog a node  
INVOL (INITIALIZE\_VOLUME) -- Initialize a disk volume (see Appendix C)  
MTVOL (MOUNT\_VOLUME) -- Mount a logical volume  
DMTVOL (DISMOUNT\_VOLUME) -- Dismount a logical volume

### **Using Miscellaneous Utilities**

CALENDAR -- Set hardware clock and calendar (see Appendix A)  
CRUCR (CREATE\_USER\_CHANGE\_REQUEST) -- Create a User Change Request form  
DCALC (DESK\_CALCULATOR) -- Evaluate logical and arithmetic expressions  
EDFONT (EDIT\_FONT) -- Edit a character font (see Appendix B)  
FSERR (FIND\_SPELLING\_ERRORS) -- Find spelling errors in a text file  
HELP -- Invoke HELP facility  
SEND\_ALARM -- Send messages to alarm servers

### **Communicating With Remote Computers**

EM3270 -- Emulate an IBM 3270 terminal  
VT100 -- VT100 terminal emulator  
VCTL (VT100\_CONTROL) -- Set/display VT100 terminal characteristics  
EMT (EMULATE\_TERMINAL) -- Emulate a dumb terminal  
TCTL (TERMINAL\_CONTROL) -- Set or display terminal (SIO line) characteristics  
SIORF (SIO\_RECEIVE\_FILE) -- Receive a file from a remote host  
SIOTF (SIO\_TRANSMIT\_FILE) -- Transmit a file to a remote host



**Chapter 4**  
**Shell Commands**



ABTSEV (ABORT\_SEVERITY\_LEVEL)

**ABTSEV (ABORT\_SEVERITY) -- Set or display the abort severity level.**

## FORMAT

**ABTSEV [options]**

Every Shell command or program returns a completion status message to its caller. The message may indicate that the program completed successfully, or it may inform its caller of a fatal internal error. Completion status messages vary in their severity. The following completion status messages appear in order of their severity:

1. OK -- the program completed successfully and performed the requested action.
2. TRUE -- the program completed successfully; its purpose was to test a condition, and the value of that condition was TRUE.
3. FALSE -- the program completed successfully; its purpose was to test a condition, and the value of that condition was FALSE.
4. WARNING -- the program completed successfully and performed the requested action. However, an unusual (but nonfatal) condition was detected.
5. ERROR -- the program could not perform the requested action because of an error in the input. The output, however, is sound.
6. OUTPUT INVALID -- the program could not perform the requested action because of a syntactic error in the input, and the output is not structurally sound.
7. INTERNAL FATAL -- the program detected an internal fatal error and stopped. The state of the output is unknown.

The ABTSEV command lets you set the severity level at which a Shell command or program aborts. If a Shell command or program meets or exceeds the abort-severity level, then it (and all of its ancestors) abort.

ABTSEV works on a per Shell program basis. A new Shell or Shell program inherits its creator's abort-severity level, and the operating system restores that abort-severity level when you exit from the Shell or when the Shell program stops.

ABTSEV is an internal Shell command.

See the PGM\_\$SET\_SEVERITY description in the *DOMAIN System Call Reference* for further information on severity levels.

## OPTIONS

Specifying ABTSEV without options displays the current abort severity level. All options must be specified in UPPERCASE letters.

**-F[ALSE]**                      Set level to false.

- W[ARNING]**           Set level to warning.
- E[RROR]**            Set level to error.
- O[UTINV]**           Set level to output invalid.
- I[NTFATAL]**         Set level to internal fatal error.
- P[GMFLT]**           Set level to program fatal error.
- M[AX\_SEVERITY]**     Set level to maximum severity error.

**EXAMPLES**

```

$ abtsev                    Show initial setting.
error
$ abtsev -W                 Set level to WARNING.
$ abtsev                    Show new level.
warning
$

```

ACL (ACCESS\_CONTROL\_LIST)

ACL (ACCESS\_CONTROL\_LIST) -- List or copy an ACL.

## FORMAT

ACL [target\_object [source\_object]] [options]

Every directory and file has an associated access control list (ACL) which lists users and their rights to the object. ACL lets you copy an ACL from one object to another, or display an ACL. For a detailed discussion of ACL structure and usage, please refer to the EDACL (EDIT\_ACL) command description.

In addition to its own ACL, each directory contains within it two additional ACLs (called "initial ACLs"): one for new files and another for new subdirectories that are created within that directory. When you create a new file or directory, the system assigns the appropriate initial ACL stored in the parent directory. Also, when you copy files or directories to new locations in the file hierarchy, the destination object receives the appropriate initial ACL from its new parent directory. To change or display these initial ACLs stored within a directory (so that newly created objects receive new initial ACLs), use the -I, -ID, or -IF options.

The ACL command only displays ACLs or copies ACLs from one object to another. To make changes to an existing ACL, use the EDACL command.

## ARGUMENTS

**target\_object**  
(optional)

Specify the object whose ACL you want to set or display. You may use a wildcard to specify this argument. DO NOT, HOWEVER, DO \$ acl /... (anything) AS THIS MAY RENDER YOUR NODE UNUSABLE. This wildcard sequence includes files in the /SYS tree, which require special ACL settings in order for system software to run.

Default if omitted: use current working directory

**source\_object**  
(optional)

Specify the file or directory whose ACL(s) is to be used to set the ACL(s) of the target object(s).

Default if omitted: display target\_object's ACL

## OPTIONS

The following options confine the ACL command's operation to target objects of the given type.

- D** Set or display ACLs of only those target objects that are directories. If used with -I, -ID, or -IF options, set or display initial ACLs for subdirectories.
- F** Set or display ACLs of only those target objects that are files.

The following options control the ACL command's effect on target objects. If the target object is a directory, they cause ACL to operate only on the initial ACLs stored within that directory for use on newly created objects, and not on the ACL of the directory itself. Note that this does NOT imply that all the target object(s) are directories, however. (That is what -D specifies.)

**-I** Set or display initial ACLs. If you are setting the ACLs of a target directory, the source object's type (file or directory) determines which initial ACL (the one for files or the one for directories) of the target directory is set. If the target object is a file (or if a wildcarded target list includes files) and the source is a directory, you will get an error unless you have also specified -IS (so that the initial file ACL in the source directory, rather than the ACL of the directory itself, can be copied to target files). If both source and target are files, then the source file's ACL is applied to the target file, as you would expect. You must run SALD (SALVAGE\_DIRECTORY) on target directories that have never contained initial ACLs (i.e., those directories created using software prior to SR4.1).

**-ID** Set or display only the initial ACLs inside those target objects that are directories that apply to new subdirectories created in those directories.

**-IF** Set or display only the initial ACLs inside those target objects that are directories that apply to new files created in those directories.

(Specifying both -ID and -IF is the same as -I. Neither implies -D.)

The following option specifies that one (or both) of the initial ACLs inside the source object is to be copied to the target, rather than the ACL of the source itself. This assumes that the source object is a directory and not a file, since files cannot contain initial ACLs for subordinate objects.

**-IS** Copy the initial ACL(s) in the source object (which must be a directory) to the target. If there is a single target object (either a file or a directory), then the appropriate initial ACL inside the source is applied to the target. If the -I option is also specified, then both initial ACLs in the source are copied to the initial ACLs inside those target objects that are directories.

The following option specifies that all the ACLs of the target object(s) are to be set or displayed.

**-ALL** Set or display all ACLs of the target object(s). If you are using wildcards to specify the target, you may qualify this action by also specifying -D or -F. If the source object is a directory, then all of its ACLs (both its own and the two initial ACLs that it applies to newly created subordinate objects) are used to set the corresponding ACLs of the target object(s). If -IS is also specified, however, the ACL of the source object itself will not be used, although all three ACLs of the target directories are still set. Thus using -ALL (with or without -IS) may be used to propagate new ACLs throughout subtrees.

## ACL (ACCESS\_CONTROL\_LIST)

The following options perform miscellaneous tasks:

- LINKS**            If `target_object` is a wildcard that specifies link(s), operate on the link(s). By default ACL does not operate on links specified with wildcards. ACL always, however, operates on links you specify explicitly (without wildcards).
- L**                List object names as the command sets ACLs.
- BR**              Display ACLs only, not object names.

ACL uses the command line parser, and so also accepts the standard command options listed in Chapter 3 with the exception of the use of hyphen (-) to read data from standard input.

### EXAMPLES

1. `$ acl new_file old_file`            Assign `old_file`'s ACL to `new_file`.
2. `$ acl joe mary -i -is`            Set the initial ACLs inside JOE using the initial ACLs inside MARY (which must be a directory).
3. `$ acl abc?* file1 -d -if`            Set the initial file ACL in all subdirectories of the current working directory whose names begin with ABC to the ACL of FILE1.
4. `$ acl abc?* dir2 -f -is`            Set the ACLs of all files in the current working directory whose names begin with ABC to the initial file ACL inside DIR2.
5. `$ acl abc?* dir2 -i -is`            The initial ACLs in all subdirectories of the current working directory whose names begin with ABC are set using the initial ACLs in DIR2, and the ACLs of all files whose names begin with ABC are set using the initial file ACL in DIR2. (Adding `-D` would confine the operation to directories.)
6. `$ acl abc?* dir2 -all`            The ACLs of all files matched are set using the initial file ACL in DIR2. The ACLs of all directories matched are set using the ACL of DIR2 itself. The initial ACLs inside those matched directories are set using the initial ACLs inside DIR2.
7. `$ acl abc?* dir2 -all -is`            The ACLs of all files matched are set using the initial file ACL in DIR2. The ACLs of all directories matched are set using the initial directory ACL in DIR2. The initial ACLs inside those matched directories are set using the initial ACLs inside DIR2.

**ARCF (ARCHIVE\_FILE) -- Maintain an archive file.**

## FORMAT

**ARCF command arcname [pathname ...]**

ARCF collects sets of files into one large file and maintains that file as an archive. Files can be extracted from the archive, new ones can be added, old ones can be deleted or replaced by updated versions, and data about the contents can be listed. Only text files can be archived.

Files to be added to an archive must exist as files with the name given. Files that are extracted from an archive will be written to files with the name given. Files that are added to archives can, of course, be archive files themselves. Any number of files can be nested this way. Thus, ARCF can be used to maintain tree-structured file directories.

**NOTE:** When you use the update and print commands, the files are updated and printed in the order they appear in the archived file, not in the order listed on the command line.

## ARGUMENTS

**command  
(required)**

Specify the operation to perform on the archive file arcname. Follow the command with V to get verbose output. Possible commands are:

- D Delete the named files from the archive. If the V option is used, filenames are displayed on the standard output as they are deleted from the archive.
- P Write the named files on standard output. The V option causes the filenames to precede the file.
- T Write a table of contents for the archive file. Normally, the table contains only the filename. If the V option is used, the table also includes the file's length, type, and date and time of the last change.
- U Update the named archive by replacing existing files, or adding new ones at the end. If no filenames are given, all possible files in the archive will be updated with files of the same name in the current directory. If the archive file does not exist, it will be created with the name given. If the V option is used, filenames are displayed on standard output as files are written to the new archived file.

## ARCF (ARCHIVE\_FILE)

**-X** Extract the named files from archive. Write each to a file with the same name. If the file already exists, the new version replaces the old. If the **V** option is added, filenames are displayed on standard output as files are extracted.

**V** Request verbose output. This command can follow any of the other commands (see example below), and will cause the archiver to print additional information, generally filenames, on standard output. Its specific action for each command has already been described.

**Arcname**  
(required)

Specify name of archive file being created or maintained.

**pathname**  
(optional)

Specify name of file to be added or deleted from the archive. Multiple names are permitted, separated by blanks. Specifying a hyphen as a filename will cause further names to be read from standard input, one per line.

Default if omitted: perform action on all files in the archive (except **-D**, which requires that names be explicitly given).

### EXAMPLES

1. 

```
$ arcf -uv my_archive stamps
stamps
$
```

 Update archive file "my\_archive" with a new copy of the file "stamps", returning verbose output.
2. 

```
$ arcf -tv my_archive
stamps      330 local  02/18/83 13:53:07
$
```

 Report on the contents of the archive.

**ARGS (ARGUMENTS) -- Echo command line arguments.**

**FORMAT**

**ARGS [-ERR[OUT]] string ...**

ARGS writes its arguments, one per line, to standard output unless -ERR is specified. Use it to write to files by redirecting standard output into a file with the ">pathname" expression. The ARGS command is useful for inserting messages and diagnostics to be reported to the display into Shell scripts and for inserting lines of text into files.

**ARGUMENTS**

**string**  
**(required)**

Specify the string of characters to be written. Multiple strings are permitted; separate strings with blanks. Strings are written one per line. To write phrases containing literal blanks, enclose strings in quotes.

**OPTIONS**

**-ERR[OUT]**

Write the string(s) to error output instead of standard output. This option is useful for writing to the transcript pad (where error output is usually directed) from an ARGS command inside a pipeline, since standard output is then connected to the pipe.

**EXAMPLES**

1. \$ args Hi there  
Hi  
there  
\$

2. \$ args "Hi there" "Mary"  
Hi there  
Mary  
\$

3. \$ args "Hi there, Mary." >my\_file

Write "Hi there, Mary." into the file MY\_FILE in the current working directory.

## BIND

### BIND -- Combine object modules into an executable file.

BIND combines two or more object modules into one executable object module. It resolves external references to global symbols and combines sections that have the same name. For full details on the binder, see the *DOMAIN Binder and Librarian Reference* manual. The binder takes the following format:

```
$ BIND pathname1 ... [pathnameN] [option]...
```

In other words, the command line simply consists of the word BIND, one or more pathnames, and zero or more options.

The binder uses the object modules stored in pathname1 through pathnameN to create an executable object file. Each pathname must be the name of a valid object file or library file. (A compiler creates an object file, and the librarian creates a library file.) You may use wildcards in pathnames. The binder automatically loads all object modules stored in object files, but conditionally loads the object modules stored in library files.

Options modify the binder's actions. Of all the binder's options, -BINARY is the most important. You must use this option to get an executable output object file.

Following is a summary of the BIND options. See the *DOMAIN Binder and Librarian Reference* manual for complete descriptions of each option. Default options are indicated by "(D)".

-ALIGN section-name LONG	Aligns the named section on a 32-bit boundary at runtime.
-ALIGN section-name QUAD	Aligns the named section on a 64-bit boundary at runtime.
-ALIGN section-name PAGE	Aligns the named section on an 8,192-bit boundary at runtime.
-ALLKEEPMARK	Preserves all marks.
-ALLMARK	Marks all global symbols in the input object files that appear after the option on the BIND command line.
-ALLRES[OLVED]	Signals a shell severity level of "error" if there are unresolved global symbols at the end of a BIND command. Useful in controlling Shell scripts.
-ALLUNMARK	(D) Unmarks all global symbols in the input object files that appear after the option on the bind command line.
-BDIR directory_name	Adds a pathname to the list of directories the binder searches for input object files.
-B[INARY] pathname	Creates an output object module and stores it at pathname.
-END	Signifies end of a command that is spread over several lines.
-ENTRY global_symbol	Specifies a nondefault start address.
-EXACTCASE	Makes the binder case-sensitive to all variable names and section names.
-GLO[BALS]	Writes currently defined global symbols to error output.
-H[ELP]	Prints this list of commands.
-INCL[UDE] module-name	Unconditionally loads the named object module from a library file into the output object file.

-INCL[UDE] -ALL Unconditionally loads all object modules from a library file into the output object file.

-INLIB pathname Specifies that the object modules in pathname are to be "installed" when the output object file is invoked. (This is an alternative to the -INLIB utility.)

-LOCALSEARCH Forces the binder to make another search through a library file if the previous search loaded an object module containing an unresolved external reference.

-LOOKS[ECTION] name Makes the named section available for sharing with a public section in an installed library.

-LOOKS[ECTION] -ALL Makes all subsequent sections available for sharing with their counterpart public sections in an installed library.

-MAK[ERS] Lists the version numbers of the compilers, binders, etc. that were used to create the input object files.

-MAP Writes a complete binder map to standard output.

-MARK global\_symbol Marks the specified global symbol.

-MARK -ALL Same as "-ALLMARK".

-MARKS[ECTION] section\_name Makes section\_name public. Affects only those object files that are destined to be installed as an installed library.

-MARKS[ECTION] -ALL Makes all subsequent sections public. Affects only those object files destined to be installed as an installed library.

-MERGE[BSS] Merges all sections corresponding to C global variables into a single section named BSS\$.

-MES[SAGES] (D) Produces informational messages at the end of a BIND command.

-MOD[ULE] new\_name Changes the name of the output object module from the default (i.e., the first input object module loaded) to new\_name.

-MSGs (D) Same as -MESSAGES.

-MULTIRES Reports errors if multiple resolutions of the same external symbol exist in object module libraries.

-NMSGs Same as -NOMESSAGES.

-NOEXACTCASE (D) Makes the binder case-insensitive to all variable and section names.

-NOINLIB pathname Specifies that the object file(s) in pathname are no longer to be "installed" when the program is invoked.

-NOLOCALSEARCH (D) Searches each library file once, then proceeds to searching the next input object file.

-NOLOOKS[ECTION] name Makes the named section unavailable for sharing.

-NOLOOKS[ECTION] -ALL (D) Makes all subsequent data sections unavailable for sharing.

-NOMARKS[ECTION] section\_name Makes section name private.

-NOMARKS[ECTION] -ALL Makes all subsequent sections private.

-NOMES[SAGES] Suppresses informational messages.

-N[O]MULTIRES (D) Omits error reporting when there are multiple possible resolutions in a library.

## BIND

<code>-NOUND [EFINED]</code>		Suppresses the listing of undefined globals.
<code>-Q [UIT]</code>		Exits from the binder without finishing.
<code>-READONLY [SECTION] section_name</code>		Changes the read/write attribute of <code>section_name</code> to read-only.
<code>-SEC [TIONS]</code>		Displays a section map.
<code>-SET_VER [SION] number.number</code>		Sets the program version in the map to the specified number.
<code>-SORTL [OCATION]</code>		Sorts global symbols numerically (by position).
<code>-SORTN [AMES]</code>	(D)	Sorts global symbols alphabetically (by name).
<code>-SYS [TEM]</code>		Makes system globals visible.
<code>-SYSTYPE type</code>		Builds a shared resource record into the output object module. For <code>type</code> , you must specify the name of an operating system ( <code>sys3</code> , <code>sys5</code> , <code>bsd4.1</code> , or <code>bsd4.2</code> ). This option overrides all system information stored in the input object modules.
<code>-UND [EFINED]</code>		Suppresses a listing of unresolved external symbols present at the end of a bind command line.
<code>-UNMARK global_symbol</code>		Remove a mark from the specified global symbol.
<code>-UNMARK -ALL</code>		Same as <code>"-ALLUNMARK"</code> .
<code>-UNMARKS [ECTION] name</code>		Makes <code>section_name</code> private. Affects only those object files that are destined to be installed as an installed library.
<code>-UNMARKS [ECTION] -ALL (D)</code>	(D)	Makes all subsequent sections private. Affects only those object files that are destined to be installed as an installed library.
<code>-XREF</code>		Displays a listing of cross references.
<code>- (hyphen)</code>		Tells the binder that more input will follow on the next line.

## EXAMPLES

- `$ bind a.bin b.bin -binary my_program` A simple binder command line. The binder builds an output object file in `my_program` from two input object files.
- `$ bind a.bin my_library -b my_program` A library file can also serve as an input object file.
- `$ bind one.bin two.bin three.bin -map -b my_program` The `-MAP` option causes bind to print substantial binder information.
- `$ BIND <RETURN>`  
`*paul.bin -ALLMARK -B name.bin <RETURN>`  
`*tine.bin -UNMARK date -UNMARK year <RETURN>`  
`*john.bin -map <RETURN>`  
`*<RETURN>` The command `BIND` specified by itself tells bind that more input will follow on the next line. Specify a blank line to end the prompting.

5. \$ BIND a.bin b.bin {a comment} -b hope Put comments inside braces.

BLDT (BUILD\_TIME)

BLDT (BUILD\_TIME) -- Display time at which operating system was built.

## FORMAT

BLDT [pathname] [options]

BLDT displays the time at which the running version of AEGIS was built.

## ARGUMENTS

**pathname**  
(optional)            Display the build time of the node whose network root directory is "pathname".  
  
Default if omitted: display build time of current node

## OPTIONS

**-N node\_spec ...**            Display build time of specified node[s]. See the section on node specifications in Chapter 3 for more information.

**-A**                            Display build time of all nodes.

## EXAMPLES

1. \$ bldt  
\*\*\*\* Node 532 \*\*\*\*    "//paris"  
AEGIS, revision 5.0, built on Friday, June 5, 1982 2:29:44 am (EST).
2. \$ bldt //os  
\*\*\*\* Node 21 \*\*\*\*    "//london"  
AEGIS1, revision 6.0x, built on Tuesday, June 24, 1983 9:01:00 pm (EDT).
3. \$ bldt -n 74  
\*\*\*\* Node 74 \*\*\*\*    "//munich"  
AEGIS1, revision 6.0x, built on Monday, July 1, 1983 10:26:41 am (EDT).
4. \$ bldt -n //brooklyn  
\*\*\*\* Node COCOA.584 \*\*\*\*    //brooklyn diskless to "//new\_york"  
AEGIS2-DOMAIN/IX kernel, revision 9.0, Friday, June 21, 1985 3:40:04 pm

**BOFF -- Deactivate the Shell's -B flag.**

**FORMAT**

**BOFF**

BOFF turns off the Shell's -B (display output of background process) flag, which is turned on by the BON command or the -B option on the SH command line. When the flag is off, the output of background processes created with the & parsing operator is sent to /DEV/NULL. This background process output is sent to /DEV/NULL by default.

BOFF requires no arguments or options.

BON

**BON -- Activate the Shell's -B flag.**

## FORMAT

### BON

BON activates the Shell's -B (display output of background process) flag. The flag can also be activated by using the -B option on the SH command when the Shell is invoked. The BOFF command deactivates the -B flag. By default, the flag is off when a Shell is invoked.

This flag causes the Shell to send the output of a background process (created with the "&" parsing operator) to the display. The output of the background process is displayed in the transcript pad of the Shell where it was invoked.

If BON is turned on in a Shell script, it remains on until that Shell script exits, or until it is over-ridden by a BOFF command in a nested Shell script. When a Shell script exits, the state of execution tracing is returned to the state in effect just before the script was invoked.

BON requires no arguments or options.

**CALENDAR (SET SYSTEM CALENDAR) -- Set system calendar clock.**

## FORMAT

**CALENDAR (from the Shell)**

**EX CALENDAR (from the Mnemonic Debugger)**

The calendar utility is used to set or reset the calendar clock in a node. It can also be used to update the last valid time known to the system, which is stored on the boot volume. Normally, the clock is set at the factory, and there is no need to reset it. Care must also be taken if setting the clock backwards in time, since duplicate UIDs may be generated, resulting in the loss of files. For information on changing the timezone, see the TZ (TIMEZONE) command description.

Note that CALENDAR only works on unmounted volumes, so you must invoke it from the Mnemonic Debugger in order to set the clock on the boot volume.

Please refer to the *DOMAIN System Utilities* manual for complete information on the use of this software tool.

CALENDAR prompts for all required arguments and options.

CATF (CATENATE\_FILE)

CATF (CATENATE\_FILE) -- Read file(s) and write to standard output.

## FORMAT

CATF [pathname ...]

CATF reads input files in order and writes them to standard output.

## ARGUMENTS

**pathname**  
(optional)

Specify file(s) to write to standard output. If multiple pathnames are given, they are read and written in the order that they appear on the command line.

Default if omitted: read standard input

## OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

1. `$ catf garbage`      Writes the file "garbage" on standard output.
2. `$ catf garbage - trash >collector`  
  
Concatenates the file "garbage," the lines read from standard input, and the file "trash," and writes the result in the file "collector."
3. `$ catf collector >>junk`  
Appends the contents of "collector" to the file "junk."



CHN (CHANGE\_NAME)

CHN (CHANGE\_NAME) -- Change an object's name.

## FORMAT

CHN old\_name [new\_name] [old\_name [new\_name] ...] [options]

CHN changes the name of a file, directory, or link. CHN works with the rightmost component ("leafname") of the old name (see EXAMPLES).

This command cannot be used to change the name of a directory embedded in a complete pathname, which would result in the file's relocation to some other part of the naming tree. For instance,

```
$ chn //et/mary/letters //et/fred/letters
```

is illegal. Use the MVF (MOVE\_FILE) command for that operation.

## ARGUMENTS

Multiple 'old\_name'/'new\_name' pairs and pathname wildcarding are permitted.

**old\_name**

(required)

Specify the current pathname of the object to be renamed.

**new\_name**

(optional)

Specify the new name of the object. The new name may be derived from the old name. 'New\_name' may be omitted entirely if -D, -Y, or -U are specified. Otherwise, some portion of it is required. Names may be 1 to 32 characters long.

Default if omitted: derive 'new\_name' from 'old\_name'.

## OPTIONS

**-D**

Append today's date (month and day) to 'new\_name' in the form "new\_name.mm.dd"

**-Y**

Append today's date (year, month, and day) to 'new\_name' in the form "new\_name.yy.mm.dd"

**-U**

Force 'new\_name' to be unique by appending a sequence number to the end of the name until it becomes unique.

**-S**

List names changed on standard output.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

1. `$ chn fritz henri` Change the name "fritz" to "henri" in the current working directory.
2. `$ chn henri mike peter paul` Change henri to mike and peter to paul.
3. `$ chn (a b c) =.zorp` Change a b and c to a.zorp, b.zorp and c.zorp.
4. `$ chn /my/stuff/lips red_lips` Change the file "lips" to "red\_lips" in the directory "/my/stuff."
5. `$ chn henri -d  
henri.07.19` Change henri to henri.mm.dd where mm is the current month (01-12) and dd is the current date (01-31).
6. `$ chn joe -u  
joe.1` Change joe by appending sequence number to end of file name.

CHPASS (CHANGE\_PASSWORD)

CHPASS (CHANGE\_PASSWORD) -- Change a login password

## FORMAT

CHPASS [new\_password]

CHPASS changes your login password to 'new\_password.' CHPASS allows you to change your password from the Shell command level.

In order for this command to work properly, the network registry must have been sealed so it is owned by the LOGIN subsystem. Refer to CRSUBS, SUBS, and ENSUBS, plus HELP PROTECTION, for more information on protected subsystems and the objects they protect.

Your login password can also be changed using the -P option under the L (LOGIN) Display Manager command.

## ARGUMENTS

**new\_password**  
(optional)

Specify new password. Omitting this option causes CHPASS to prompt for your new password. Input echo is disabled. A second prompt verifies the password and guards against typing mistakes.

Default if omitted: prompt up to three times for password.

## EXAMPLES

```
$ chpass sesame      Set new login password to "sesame."
```

**CHPAT (CHANGE\_PATTERN) -- Replace pattern in text file.**

## FORMAT

**CHPAT [options] [pathname ...] [-P] [pat ...] from\_pattern [to\_expression]**

CHPAT copies every line from its input files to its output files, globally substituting the text replacement pattern "to\_expression" for each occurrence of "from\_pattern" in those lines designated by the "pat" argument(s) and any options.

Refer to the descriptions of the ED (EDIT), FPAT (FIND\_PATTERN), and EDSTR (EDIT\_STREAM) commands for related information.

## ARGUMENTS

**pathname**  
(optional)

Specify name of file to be searched. Multiple pathnames and wildcarding are permitted.

Default if omitted: search standard input.

**from\_pattern**  
(required)

Specify target text string (a regular expression) for substitution or deletion. If the string includes the characters % \$ [ ] { } ! \* or any other Shell special characters, enclose it in quotes to avoid unpredictable results. If the 'pathname' argument is present, precede this argument (or the 'pat' argument, if present) with "-P" to separate the pathname(s) from the regular expressions on the command line.

**to\_expression**  
(optional)

Specify replacement string. If no replacement is specified, the 'from\_pattern' string will be deleted. If regular expressions defining a range of text ('pat' argument) are present, you must use a literally null 'to\_expression' ("") to delete 'from\_pattern'.

**pat**  
(optional)

Specify range of text for which the substitution is to apply, in the form of a regular expression. Multiple expressions separated by blanks are permitted. Unless modified by options, any line of text matching any pattern is replaced and all lines, changed or not, are written to output. If the 'pathname' argument is present, precede this argument with "-P" to separate the pathname(s) from the regular expressions on the command line.

Default if omitted: use 'from\_pattern' to select matching lines.

## CHPAT (CHANGE\_PATTERN)

### OPTIONS

- A**                   Select only lines that match all of the leading expressions, in any order.
- X**                   Select only lines that match none of the leading expressions.
- O**                   Write only the selected lines to standard output. By default, CHPAT writes all lines to output.
- L**                   List name(s) of input file(s) on output file(s) as the input file(s) are searched.
  
- OUT pathname**       Specify name of output file. Pathname may be derived from the input file name. If this option is omitted, output is written to standard output.

### EXAMPLES

1. `$ chpat foo bar`

Changes all occurrences of 'foo' in standard input to 'bar' and writes the results to standard output.

2. `$ chpat '%This' " *" ' '`

In lines starting with 'This', it changes all occurrences of multiple spaces to a single space.

3. `$ chpat '%This' '%That' " *" ' '`

Like 2., but works on lines starting with either 'This' or 'That'.

4. `$ chpat -a '%when' 'only$' ';' ':'`

In lines that start with 'when' and end with 'only', change all semicolons to colons.

5. `$ chpat -x not none some all`

In lines that do not contain either 'not' or 'none', change all instances of 'some' to 'all'.

6. `$ chpat erase`

Delete (replace with nothing) all occurrences of 'erase'. Exactly the same effect can be obtained with:

```
$ chpat erase ''
```

7. `$ chpat -o other_opts pat... fr_pat to_pat`

Is exactly the same as

`$ fpat pat... | chpat other_opts fr_pat to_pat`

8. `$ chpat ?*.pas -out =.new -p "if x = y" "if (x = y)"`

Change all occurrences of the string "if x = y" to "if (x = y)" in all Pascal source files (files ending with '.pas') and put the output for 'X.pas' in 'X.pas.new'.

CMACCT (COMPARE\_ACCOUNT\_FILE)

CMACCT (COMPARE\_ACCOUNT\_FILE) -- Compare account files.

## FORMAT

**CMACCT rgy1\_path rgy2\_path**

CMACCT compares two different network registries' accounts. Accounts that appear in both registries and are associated with different home directories and/or passwords are reported. Use EDACCT to resolve these account collisions.

CMACCT is for use in a DOMAIN/BRIDGE Internet. For complete information on comparing network registries, see *Managing DOMAIN Internets*.

## ARGUMENTS

**rgy1\_path**  
(required)

Specify the pathname of a registry file to be compared. The pathname may be the master registry file (//node/REGISTRY/RGY\_MASTER), a node's copy of the master registry file (//node/REGISTRY/REGISTRY), or simply a node entry directory (//node). If the pathname is a node entry directory, the //node/REGISTRY/REGISTRY file is used.

**rgy2\_path**  
(required)

Specify the other registry file to be compared.

## EXAMPLES

The following commands compare the account files in two different registries, delete the account that is associated with different home directories from one of the registries, and then compare the account files again to check that there are no more collisions.

```
$ cmacct //alpha/registry/rgy_master //beta/registry/rgy_master
```

```
Account user.none.none has different home directories: / //guest/user
```

```
1 account collision
```

```
$ edacct -r //alpha/registry/rgy_master
```

```
=> d user none none
```

```
user none none /
```

```
OK to delete this account? (Y/N/Quit): y
```

```
Current entry is: smith none r_d //martha/smith
```

```
=> wr
```

```
$ cmacct //alpha/registry/rgy_master //beta/registry/rgy_master
```

```
No account collisions
```

```
$
```

**CMF (COMPARE\_FILE) -- Identify differences among files.**

## FORMAT

**CMF file\_a [...file\_e] [options]**

CMF compares the contents of two to five ASCII files and reports the differences on standard output. This command works only on files: to compare directory trees, use CMT (COMPARE\_TREE).

## ARGUMENTS

**file\_a**

**(required)**

Specify pathname of original file; all differences are reported in relation to this file. Wildcarding of this pathname is permitted to achieve multiple comparisons.

**file\_b ... file\_e**

**(optional)**

Specify descendants of 'file\_a'. If more than one file is specified, you may use a hyphen (-) to cause standard input to be read in place of a pathname. Pathnames may be derived from 'file\_a'.

Default if omitted: read standard input

## OPTIONS

**-R pathname**

Report all differences to the specified report file, in addition to reporting to standard output. This pathname may be derived from 'file\_a' (if 'file\_a' is wildcarded) to produce one report for each comparison. If 'file\_a' is wildcarded and this report file name is NOT derived, then all reports are concatenated into the single report file.

**-TB**

Include trailing blanks in the comparison. By default, ignore trailing blanks. -TB also causes CMF to regard the NEWLINE character at the end of the last line in the file as significant (if it exists).

**-BR**

Display only line numbers of lines containing discrepancies. By default, display both line numbers and line contents.

**-L**

Display names of files being compared before each comparison is performed. This is useful when wildcarded pathnames are specified.

**-M n**

Set the minimum number of lines for a rematch to 'n.' This is the minimum number of lines, following a reported difference, which must match for CMF to consider the files synchronized. The default value is 3.

## CMF (COMPARE\_FILE)

### EXAMPLES

Assume that file "file1" contains

```
Fourscore  
and seven  
years ago,  
our fathers brought  
forth
```

and "file2" contains

```
Eighty-seven  
years ago,  
our fathers brought
```

CMF produces the following output when "file1" is compared to "file2."

```
$ cmf file1 file2  
  
A1      Fourscore  
A2      and seven  
changed to  
B1      Eighty-seven  
  
A5      forth  
deleted before  
end of file B  
  
2 discrepancies found.
```

**CMPPO (COMPARE\_PPO\_FILE) -- Compare person, project, or organization names.**

## FORMAT

**CMPPO rgy1\_path rgy2\_path [options]**

CMPPO compares two different network registries' person, project, and/or organization names. Names that appear in both registries and are associated with different name IDs are reported. Use the C (CHANGE\_NAME) command in EDPPO to change duplicate names in one of the registries; do *not* delete names.

CMPPO is for use in a DOMAIN/BRIDGE internet. For complete information on comparing network registries, see *Managing DOMAIN Internets*.

## ARGUMENTS

**rgy1\_path**  
(required)

Specify the pathname of a registry to be compared. The pathname may be the master registry file (//node/REGISTRY/RGY\_MASTER), a node's copy of the master registry file (//node/REGISTRY/REGISTRY), or simply a node entry directory (//node). If the pathname is a node entry directory, the //node/REGISTRY/REGISTRY file is used.

**rgy2\_path**  
(required)

Specify the other registry to be compared.

## OPTIONS

Default options are indicated by "(D)."

- PERS** Compare the names in the Person files.
- PROJ** Compare the names in the Project files.
- ORG** Compare the names in the Organization files.
- ALL (D)** Compare the names in the Person, Project, and Organization files.

## EXAMPLES

The following commands compare person names in two registries, change duplicate names, and then compare person names again to check that all duplicate names have been handled.

```
$ cmppo //alpha/registry/rgy_master //beta/registry/rgy_master -pers
```

Names with different ids found in both registries' person file:

```
martin      smith
```

```
2 names.
```

## CMPPPO (COMPARE\_PPO\_FILE)

```
$ edppo -r //beta/registry/rgy_master -pers  
=> c martin emartin  
=> c smith msmith  
=> wr
```

```
$ cmppo //alpha/registry/rgy_master //beta/registry/rgy_master -pers
```

No names with different ids found in both registries' person file

```
$
```

**CMSRF (COMPARE\_SORTED\_FILE) -- Find lines common to two files.**

## FORMAT

**CMSRF [options] file1 [file2]**

CMSRF reads sorted files, 'file1' and 'file2', and produces 1-, 2-, or 3-column output. Column 1 contains lines found only in 'file1', column 2 contains lines found only in 'file2', and column 3 contains lines found in both files. The number option, -N, specifies which columns you want to print. To compare unsorted files, use CMF (COMPARE\_FILE).

## ARGUMENTS

Use of a hyphen for either file name will cause the data to be read from standard input.

**file1**  
(required) Specify first file for comparison.

**file2**  
(optional) Specify second file for comparison.  
Default if omitted: compare file1 to standard input.

## OPTIONS

If no options are specified, CMSRF produces a complete 3-column report.

**-n** Specify number(s), where n is an integer sequence representing the following:

- 1 Report only lines exclusive to 'file1'.
- 2 Report only lines exclusive to 'file2'.
- 3 Report only lines common to both files.

## EXAMPLES

1. `$ cmsrf -12 //us/sorted_stuff.c` Compare '//us/sorted\_stuff.c to standard input and report lines found in either place, but not both.

2. `$ cmsrf -3 //us/sorted_stuff.a //us/sorted_stuff.b`

Report only common lines for both files.

CMT (COMPARE\_TREE)

CMT (COMPARE\_TREE) -- Compare source tree to target tree.

## FORMAT

CMT *source\_pathname target\_pathname [options]*

CMT compares all the objects in the source tree against all objects in the target tree. CMT reports any objects cataloged in the source that do not also appear in the target. If the target contains objects that do not appear in the source, however, the differences are ignored.

CMT compares objects based on their internal representation, unlike CMF (COMPARE\_FILE), which treats its input data as ASCII text streams and compares them as such.

Both the source and target pathnames must specify the same type of object, either a directory or a file. CMT, however, can compare objects of any type, unlike CMF, which compares only text files.

If CMT encounters differences, it reports that the objects are different and continues the comparison with the next object.

## ARGUMENTS

Both the source and target pathnames must specify the same type of object, either a directory or a file. Use of wildcards in pathnames is permitted. Multiple source/target pairs are permitted.

**source\_pathname**  
(required)           Specify source tree.

**target\_pathname**  
(required)           Specify target tree. Name may be derived from  
'source\_pathname.'

## OPTIONS

If no options are specified, CMT will only report the names of directories and files with differences in source and target trees.

**-L**                   List all directories and files compared.

**-LD**                  List all directories compared.

**-LF**                  List all files compared.

**-AE**                  Abort on the first mismatch, or if the source tree contains a name not found in the target tree. By default, the comparison continues after the mismatch is reported.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### EXAMPLES

Assume that the directories "dir1" and "dir2" each contain three files called "a," "b," and "c," and that the contents of the "b" files differ. Following is the result of a comparison of those two directories.

```
$ cmt dir1 dir2
*** compare failed at file loc 0 SRC: 10002 DST: 100011
dir1/b - compare failed (from US / file utility)
```

CPBOOT (COPY\_BOOT)

CPBOOT (COPY\_BOOT) -- Copy the system boot file SYSBOOT.

## FORMAT

CPBOOT *source\_directory* { *target\_directory* | -DEV CT }

CPBOOT copies the system boot file SYSBOOT from one directory to another. The sysboot file is used by the bootstrap PROM to start the system. CPBOOT is useful for copying SYSBOOT to a floppy disk, thus making the stand-alone utilities (SAU) directory on the floppy disk accessible from the boot PROM. You may also use it to update a Winchester disk when a new software release is distributed.

If you wish to build a bootable cartridge tape, "-DEV CT" should be specified in place of the target directory. This will copy CTBOOT -- the cartridge tape version of SYSBOOT -- from the source directory (usually /SYS) onto the beginning of the cartridge tape. (Note: subsequent WBAKs to the tape should use the -SYSBOOT option to avoid overwriting CTBOOT on the tape.)

## ARGUMENTS

**source\_directory**  
(required)

Specify directory containing the file SYSBOOT or CTBOOT.

**target\_directory**  
(optional)

Specify directory to which SYSBOOT is to be copied. This must be the entry directory on the target logical volume.

Default if omitted: must use -DEV.

## OPTIONS

**-DEV CT**

Specify that you wish to build a bootable cartridge tape. This option must be specified if you omit the 'target\_directory' argument.

## EXAMPLES

```
$ cpboot /flpa /      Copy the SYSBOOT file from the directory  
                      "/flpa" to the current node entry directory.
```

```
$ cpboot /sys -dev ct Copy the CTBOOT file from the directory "/sys"  
                      to the cartridge tape.
```

CPF (COPY\_FILE) -- Copy a file.

## FORMAT

CPF **source\_pathname** [**target\_pathname**] [**options**]

CPF copies a file from the source pathname to the target pathname. CPF only copies files; see CPT (COPY\_TREE) for copying directories and their subordinate objects.

## ARGUMENTS

**source\_pathname**  
(required)

Specify file to be copied. If the source pathname is a link name, CPF resolves the link and copies the file to which the link refers.

**target\_pathname**  
(optional)

Specify target for copy. If **target\_pathname** is a directory, **source\_pathname** is copied into this directory. Target must NOT be a link.

Default if omitted: copy 'source\_pathname' into current working directory

Multiple source/target pairs and pathname wildcarding are permitted.

## OPTIONS

Default options are indicated by "(D)."

- C (D) Create source file at target. An error occurs if the target file already exists.
- R Replace target with copy of source.
- LF List files copied.
- LDL List files deleted as a result of a "replace" (-R).
- CHN Use with -C to change the name of an existing object with the target pathname before the copy is made. Use with -R to change the name of a target file if it is in use and cannot be deleted.
- DACL (D) Apply the target directory's default ACL for files copied. In addition to its own ACL, each directory has two default ACLs, one for its files and another for its subdirectories. Unless 1) you specify -SACL, or 2) the file belongs to a protected subsystem (see -SUBS), the system assigns to the target file its parent directory's default ACL for files.
- SACL Retain the source file's ACL.
- SUBS (D) Retain source ACL for objects which belong to subsystems.

## CPF (COPY\_FILE)

- NSUBS** Apply the target directory's default ACL for objects which belong to subsystems.
- F** Force deletion of target object if 'p' rights are present.
- DU** Delete when unlocked. This option is useful with **-R**. If the object to be replaced is locked when CPF is invoked, the replace operation will be performed when the object is unlocked.
- PDT** Preserve the source file's modification and used times.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### EXAMPLES

1. `$ cpf /latest/wbak wbak.latest`  
`$` Copy the file "wbak" from the "/latest" directory to the current directory, and call it "wbak.latest".
2. `$ cpf /latest/com/wbak /com -r`  
`$` Copy the file "/latest/com/wbak" to the "/com" directory, replacing the existing "/com/wbak".
3. `$ cpf /games/space?* -lf`  
(file) "spacewar" copied.  
(file) "spacebunny" copied.  
(file) "space\_shot" copied.  
`$` Copy and list all files in the "/games" directory starting with "space" to the working directory.
4. `$ cpf ?*.pas backup/= .12.07`  
`$` Copy all files in the working directory with the suffix ".pas" to the directory "backup", appending a date.

**CPL (COPY\_LINK) -- Copy a link.**

## FORMAT

**CPL linkname [pathname] ... [options]**

CPL copies a linkname to the target object.

## ARGUMENTS

Multiple linkname/pathname pairs and wildcarding are permitted.

**linkname**

**(required)**

Specify the name of the link to be copied.

**pathname**

**(optional)**

Specify the target pathname of the copied link. If 'pathname' is a linkname, then this link is created or replaced (depending on various options below). If 'pathname' is a directory, then the link text is copied into this directory. In no case is the object to which the link refers affected: only the text of the link itself.

Default if omitted: copy link into current working directory.

## OPTIONS

Default options are indicated by "(D)."

- C**            **(D)**    Create source link at target. An error occurs if the target link already exists.
- R**                        Replace target with copy of source.
- LL**                        List links copied.
- LDL**                       List links deleted because of replacement (-R).
- CHN**                       Change name of existing link with target\_pathname before copying.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

1. `$ cpl //ai/sources /progs`            Copy the link "//ai/sources" to the node entry directory as "progs".

## CPL (COPY\_LINK)

2. `$ cpl //zorba/sys/print /sys -r` Copy the link `"/sys/print"` from the node whose entry directory is `"zorba"` to the local `/sys` directory, replacing any existing link.

**CPSCR (COPY\_SCREEN) -- Copy the current display to a file.**

## FORMAT

**CPSCR** *pathname* [-INV] [-APPEND] [-GPR[\_BITMAP]]

CPSCR copies the current screen image (without clearing it) to the file you specify. Use the PRF (PRINT\_FILE) command to print the file.

Use the Display Manager command CPO to copy the screen without creating a new process window or changing the current transcript pad. CPO invokes the CPSCR command from the Display Manager without creating a pad or window. Thus, press <CMD> then type:

CPO /COM/CPSCR *pathname*

You may copy small portions of a black and white screen (such as a single window) with the DM command XI.

By default, black and white screens are copied into a GMF file. Color screens are copied into a GPR bitmap.

## ARGUMENTS

<b>pathname</b> (required)	Specify file to which the screen is copied.
-------------------------------	---

## OPTIONS

<b>-INV</b>	Invert image. Use this option to store the image in reverse video. Black screen pixels become white and white screen pixels become black. This switch cannot be used with the -gpr_bitmap switch or on color nodes.
-------------	---

<b>-APPEND</b>	Appends a black and white screen image to an existing GMF file. This switch cannot be used with the -gpr_bitmap switch or on color nodes.
----------------	---

<b>-GPR[_BITMAP]</b>	Use this option to copy a black and white screen into a gpr bitmap file rather than a gmf file. This option has no meaning for color nodes since color screens are already copied into gpr bitmaps.
----------------------	---

## EXAMPLES

1. `$ cpscr //us/looky_there -inv` Invert and copy the current screen image to the specified file. Since the command line is echoed in the Shell's process transcript pad prior to execution, this command will appear in the resulting image.

## CPSCR (COPY\_SCREEN)

### 2. <CMD>

Command: `cpo /com/cpscr //us/looky_there -inv`

Same result as in example 1,  
but the CPSCR line will not  
appear in the plotted output.



CPT (COPY\_TREE)

CPT (COPY\_TREE) -- Copy a directory tree.

## FORMAT

CPT *source\_pathname target\_pathname ... [options]*

CPT is a multipurpose tool for copying, merging, and replacing files, directories, and links. To copy files only, use CPF (COPY\_FILE).

## ARGUMENTS

Multiple source/target pairs and wildcarding are permitted.

**source\_pathname**  
(required)

Specify the file, link, or directory tree to be copied. CPT does not change the contents or link references of the source, so errors that occur will leave the source unaffected.

**target\_pathname**  
(required)

Specify the file or directory tree to be created, replaced, or merged. The target pathname may be derived from the source pathname. The target can NOT be a link. In addition, the target can NOT be a logical volume entry directory, or the network root unless the -MD option is specified.

## OPTIONS

Default options are indicated by "(D)."

**-AF date** Copy only objects whose dtms (date-times) are after the given date and time: [[[yy]yy/]mm/dd][.][hh:mm[:ss]] | TODAY. The date defaults to today, and the time to midnight; if either are omitted from 'date'.

**-BE date** Copy only objects whose dtms are before the given date and time: [[[yy]yy/]mm/dd][.][hh:mm[:ss]] | TODAY. The date defaults to today, and the time to midnight if either are omitted from 'date'.

**-C (D)** Create source at target. If the file or directory already exists, an error will occur and processing will continue to the next source/target pair. Not valid if -MS, -MD, or -R is specified.

If the source is a file, CPT copies it to the target. If the source is a directory, CPT copies the directory to the target. It then copies every file cataloged in the directory (and all subdirectories) until it reaches the end of the tree.

Each link name in the source tree is created as a link name in the target, but the object that the link references is not copied. If 'source\_pathname' is itself a link, however, the link is resolved and the object to which it points is copied to the target.

- R** Replace target with source. Not valid if **-C**, **-MS**, or **-MD** is specified. CPT deletes the tree starting at the target pathname and copies the entire source tree in its place. Note that the target is deleted BEFORE copying begins. If no target tree by the specified name exists, CPT creates one and duplicates the source.
- MS** Merge source and target if both are directories. Not valid if **-C** or **-R** is specified. If the target does not exist, CPT duplicates the source at the target. If the target exists, CPT merges the source into the target, replacing files and links, and combining directories.
- If both the source and the target are directories, CPT merges their contents as described below. Otherwise, CPT deletes the target and replaces it with the source.
- To merge directories, CPT compares their contents, object by object. Objects that exist in the source but not in the target are created in the target. Objects that exist in the target but not in the source remain unchanged. Files and links with the same name in both the source and the target are deleted from the target and replaced by the source version. Directories with the same name in both source and target are merged. CPT continues this process recursively until it reaches the end of the source tree.
- MD** Merge source and target if both are directories. Similar to **-MS** except that files and links with the same name in both source and target are left unchanged in the target.
- F** Force deletion of target object if 'p' rights are present.
- DACL** (D) Apply the target directory's default ACLs. In addition to its own ACL, each directory has two default ACLs, one for its files and another for its subdirectories. **-DACL** causes CPT to apply the target directory's default ACLs to each subdirectory and file it copies. The **-SACL** option causes each object to retain its original ACL.
- SACL** Retain the source ACL.
- CHN** Use with **-C** to change the name of a target before source is copied. Use if target\_name already exists. Use with **-R**, **-MS**, and **-MD** to change the target\_name if target is in use.
- SUBS** (D) Retain source ACL for objects which belong to subsystems.
- NSUBS** Apply the target directory's default ACL for objects which belong to subsystems.
- PR pathname** Preserve the object 'pathname' in target when another object with the same name exists in the source. Valid with **-MS** option only.

## CPT (COPY\_TREE)

**-PDT** Preserve the source's modification and used times.

The following five options allow you to monitor CPT's operation. You can use **-LD**, **-LF**, and **-LL** in any combination. By default, the listing options apply to both copied and deleted objects. To list only deletions, use **-LDL** with **-L**, **-LD**, **-LF**, or **-LL**.

**-L** List all objects as they are copied.

**-LD** List directories as they are copied.

**-LF** List files as they are copied.

**-LL** List links as they are copied.

**-LDL** List only objects deleted as a result of replacements.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### WARNING

Five conditions always terminate execution:

- You attempt to use the network root or node entry directory as a target, without specifying a merge.
- An error occurs in reading the top level of the source tree.
- You attempt to create an existing directory (if the target is an existing directory, you must specify **-R** or **-M**).
- The logical volume containing the target directory is full.
- A quit or stop fault occurs in this process.

### EXAMPLES

1. `$ cpt /com /com.backup -r` Copy the directory tree `"/com"` to `"/com.backup"` replacing the existing `"/com.backup"` tree.
2. `$ cpt my_circuits /circuits -ms` Merge the directory tree `"my_circuits"` into the `"/circuits"` tree.

**CRD (CREATE\_DIRECTORY) -- Create a directory.**

### FORMAT

**CRD** *pathname* ...

CRD creates a directory with the specified pathname.

### ARGUMENTS

**pathname**  
(required)

Specify the subdirectory name to be created. Multiple pathnames are permitted. The new directory receives its parent directory's initial ACL.

### OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### EXAMPLES

```
$ crd /my_dir/new_dir  Create the subdirectory new_dir
                        in the directory /my_dir.
```

CREFS (CROSS\_REFERENCE\_SYMBOLS)

CREFS (CROSS\_REFERENCE\_SYMBOLS) -- Cross-reference symbols in a file.

## FORMAT

CREFS [-F] [pathname ...]

CREFS produces a cross-referenced list of the symbols in each of the named files, and writes each list to standard output. A symbol is a string of letters, digits, underscores, and dollar signs and must begin with a letter. The list contains every symbol in the file in alphabetical order, followed by the numbers of the lines in which the symbol appears.

Symbols of more than 32 characters are truncated.

## ARGUMENTS

**pathname**  
(optional)

Specify input file. Multiple pathnames and wildcarding are permitted: separate names with blanks.

Default if omitted: read text from standard input.

## OPTIONS

If the option is not specified, CREFS treats uppercase and lowercase letters as different characters, and places uppercase letters before lowercase letters in the alphabetical sort.

**-F** Treat all input text as lowercase while cross-referencing.

**-K key\_file** Only the words listed in 'key\_file' are cross-referenced. These words must be listed one per line.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

To find all occurrences of certain variables in the program "cycle," type:

```
$ crefs cycle
```

CREFS can also be used in conjunction with other commands to produce more refined results. For instance:

```
$ crefs cycle | tee cycle.all | fpat wheel spoke axle >cycle.some
```

The output file "cycle.all" contains a list of all the symbols in the program, with references to the line containing them. The output file "cycle.some" contains only the lines with references to the three variables named: wheel, spoke, and axle.

**CRF (CREATE\_FILE) -- Create a file.**

## FORMAT

**CRF pathname...**

CRF creates a zero-length file with the specified pathname. The new file receives its parent directory's initial ACL for files. There is a maximum of 1300 files per directory.

(See the ACL command description for more information.) The file type is set to OBJECT and the file is made permanent. The type UID is set to nil.

This command is most useful for system-level debugging. Use the Display Manager editing capability to create normal text files.

## ARGUMENTS

**pathname**  
(required)

Specify file to be created. Multiple pathnames are permitted, separated by blanks.

## OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ crf my_file
```

CRL (CREATE\_LINK)

CRL (CREATE\_LINK) -- Create a link.

## FORMAT

**CRL linkname object\_name ... [-R]**

CRL is used to create links. Links normally serve two functions: as a shorthand way of specifying objects with lengthy (and frequently recurring) pathnames and as static pointers to other objects.

Links cause the Shell to redirect a pathname to another object. In effect, links allow you to take a detour from one part of the naming tree to another.

## ARGUMENTS

**linkname**  
(required) Specify the link's name and location.

**object\_name**  
(required) Specify the object to which the link points.

Multiple linkname/pathname pairs are permitted.

## OPTIONS

**-R** Replace an existing link. Use this option to change a link's object\_name.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ crl bugs /maintenance/reports    Create a link called "bugs" in  
                                     the current working directory.
```

Now, when you use "bugs" in a pathname, the command Shell substitutes the text "/maintenance/reports". Therefore, the pathname:

```
bugs/sticky_cursor
```

refers to the same file as the pathname:

```
/maintenance/reports/sticky_cursor
```

**CRP (CREATE\_A\_PROCESS) -- Create a process on a remote node.**

## FORMAT

**CRP [command line] -ON node\_spec [options]**

CRP creates a process on a remote node.

## ARGUMENTS

**command line**  
**(optional)**

Specify command line to be executed by the remote process. If the command string contains embedded blanks, enclose it in quotation marks.

Default if omitted: execute /COM/SH (the Shell)

## OPTIONS

The following option, which specifies the remote node, is **REQUIRED**:

**-ON node\_spec**

Specify the remote node on which the process is to be created. See the section on node specifications in Chapter 3 for more information.

One of the following options may be specified (default option is indicated by a "(D)"):

**-CP (D)**

Create a remote process running with standard streams connected to the current window. Not valid if -CPO or -CPS is specified. You may stop these processes by typing an EEF (usually CTRL/Z) in the process input pad.

**-NWP**

Do not create a window pane legend indicating that the local window is connected to a remote process. Use with -CP option only.

**-CPO**

Create a remote process without a connection to the current window, and an identity of 'user.none.none'. Not valid if -CP or -CPS is specified. To stop these processes, you must first create a visible remote process running the Shell, then issue the SIGP command to stop the background process.

**-CPS**

Create a remote process without a connection to the current window, and an identity of 'user.server.none'. Not valid if -CP or -CPO is specified. To stop these processes, you must first create a visible remote process running the Shell, then issue the SIGP command to stop the background process.

**-N name**

Specify the name of the remote process. If this option is not specified, the default is "user id.node\_id". This allows remote processes to be traced to their originator.

**-LOGIN name [password]**

Specify the login sequence for the remote process on the command line. If the password is omitted, the system prompts you for it. A null (zero-length) password is specified by the null string "".

Normally, -LOGIN appears with -CP. You may, however, use -LOGIN with -CPO and -CPS as well. If -LOGIN is specified with either -CPO or -CPS, then the identity of the created process is the same as that of the caller's (as opposed to 'user.none.none' or 'user.server.none', respectively). This means that -CPO and -CPS are identical if -LOGIN is also specified.

If you use -LOGIN with -CPO or -CPS, you must place both the name and the password on the command line. No prompting is available in this case.

**-ME**

-ME is typically specified instead of -LOGIN. If -ME is specified, then the created process on the remote node inherits the caller's working directory, naming directory, home directory text string and SID. In some sense, this is similar to popping up another Shell except that the process is running on another node. If -ME is specified with either -CPO or -CPS, then the identity of the created process is also that of the caller's (as opposed to 'user.none.none' or 'user.server.none', respectively). This means that -CPO and -CPS are identical if -ME is also specified.

**-QUIET**

Suppress connection/disconnection messages in the transcript pad.

**EXAMPLES**

\$ crp -on 532 -login joe      Create a process on node 532 running the Shell, and log in with the user id "joe".

\$ crp -on 0aef -me            Create a process on node AEF running the Shell, and inherit the current process state information.

**CRPAD (CREATE\_PAD) -- Create a transcript pad and window.**

## FORMAT

**CRPAD [pathname] [options]**

CRPAD creates a transcript pad, copies a file (or standard input) into that pad, and then opens a window into the pad. This new pad is NOT related to the transcript pad attached to processes running the Shell; it is for viewing file contents only. This is primarily useful for displaying output being produced inside a pipeline without interrupting the flow of control in the pipe.

Transcript pads are not editable. If you wish to place a file in a pad for editing, use the <EDIT> key or the DM command CE (CREATE\_EDIT).

CRPAD -IN behaves differently. This creates an edit pad and lets you create whatever text you want. When you close the edit pad (with WC or CTRL/Y), that text is copied to standard output.

## ARGUMENTS

**pathname**  
(optional)

Specify the file to be copied into the pad. Not valid if -IN is present.

Default if omitted: copy standard input.

## OPTIONS

**-IN[PUT]**

Copies data from a temporary edit window to standard output. Not valid if -TEE or -PN are specified.

**-PN pathname**

Specify a pathname for the pad. If you specify a pathname, the pad is saved in that file. Note that you can also save the pad after it is created by using the DM command PN (PAD\_NAME).

**-TEE**

Copy output to standard output in addition to the new pad.

## EXAMPLES

1. `$ crpad test.data`      Create a pad that displays the file "test.data".
2. `$ fpat -p '256-' <phone.book | crpad -tee | srf >phone.book.local`  
Display the intermediate results in a pipeline.
3. `$ crpad -input | srf | crpad`  
Create an edit pad. When the pad is closed, sort the text edited and display it in a transcript pad.

CRRGY (CREATE\_REGISTRY)

CRRGY (CREATE\_REGISTRY) -- Create or modify network registry.

## FORMAT

**CRRGY [options]**

CRRGY allows you to create a network registry, create a local registry on the master node, or change registry sites. While all the functions of this command are described below, it is unlikely that you will be able to manipulate the network registry unless you are the network administrator for your network (due to ACL settings on the registry files). You may, however, have access to the local registry for your particular node to manipulate as you like. For complete information about both network and local registries, see *Administering Your DOMAIN System*.

## OPTIONS

At least one of the following options must be specified.

**-R rgypath** Specify the pathname of the new registry object you want to create or the registry on which you want to operate. If you provide just a node name (//node), CRRGY creates the registry "//node/registry/rgy\_master", or CRRGY uses the file //'this\_node'/registry/registry to locate the master registry. If you omit a pathname, CRRGY creates "//'this\_node'/registry/rgy\_master." We recommend that you specify only a node name.

Default if omitted: create //'this\_node'/REGISTRY/RGY\_MASTER

**-S site...** Create new site(s). 'site' specifies the pathname of the registry directory you want to create. Multiple sites are permitted, separated by blanks. Files created at the sites are filled with default identities. The new sites are included in the new registry master file. If you do not specify -S, and you are creating a registry, "//'this\_node'/registry/rgy\_site" is used. Sites may be located anywhere in the network. This option is not valid for local registries.

**-A site...** Add site name(s) to the existing registry specified by the -R option. This command creates a directory at the site, copies all the registration data files to the directory, and adds the site names to the master file. This option is not valid for local registries.

**-D site...** Delete site name[s] in the existing registry master file specified by the -R option. The master registry file reflects the change but the registry file copies ('/registry/registry') do not. This option is not valid for local registries. The site directories are NOT deleted; their names are removed from the registry file.

- LOC** Create a local registry. The only valid pathnames for a local registry are the following default names:
- "/'/node'/registry/local\_registry" for the master, and
- "/'/node'/registry/local\_site" for the site.
- Thus, you should only specify a root name ("/'/node'") with the -R option and let all other defaults apply.
- EX** Specify that the given sites already exist. Do not create the sites specified, just include their names in the registry master file. You may use this option to create a replacement registry master file that points to existing sites or to add existing site names to an existing master file (-A). No check is made to ensure that the sites really do exist. The site names must be full pathnames when you use this option.
- SEC** Set secure passwords to a minimum length of six characters. This option is not valid for local registries.
- COUNT n** Specify the number of slots (n) to create in a local registry account file. The default number is 25; minimum is 5. This option requires -LOC. Nodes serving diskless partners should have enough entries to allow slots for the users of the diskless nodes.
- DAYS n** Specify number of days (n) during which an entry in a local registry account file will remain valid after login. The default number is 21; maximum is 60. This option requires -LOC.

**EXAMPLES**

```
$ crrgy -loc -count 50 -days 30      Create a new local registry with
$                                     spaces for 50 account entries
                                       which will each be valid for
                                       30 days after login. Previous
                                       local registry accounts are lost.
```

For detailed examples of network registry creation, see *Administering Your DOMAIN System*.

CRSUBS (CREATE\_SUBSYSTEM)

CRSUBS (CREATE\_SUBSYSTEM) -- Create a protected subsystem.

## FORMAT

**CRSUBS subsystem\_name**

CRSUBS is used to create a protected subsystem.

A protected subsystem is a set of programs and data files. The programs are called the "managers" of the protected subsystem; the data objects which they manage are said to be "owned" by the subsystem. Protected subsystems allow you to define exactly how a file can be accessed.

CRSUBS creates a protected subsystem by creating the "subsystem shell" and putting it in the directory /SYS/SUBSYS. Once there, the subsystem shell can be invoked using the ENSUBS command. The access control list for the directory /SYS/SUBSYS determines who can create new subsystems: whoever has 'a' (append) rights to the directory will be able to create new subsystems.

The access control list on the file /SYS/SUBSYS/subsystem\_name determines who can enter the subsystem 'subsystem\_name'. Whoever has read and execute rights to it can enter the subsystem. This capability should be restricted generally to the creators of the subsystem or to the system administrators.

For more information on protected subsystems, see the *DOMAIN System User's Guide*.

## ARGUMENTS

**subsystem\_name**  
(required)

Specify name of subsystem to be created. This file will contain the "subsystem shell" and will reside in the /SYS/SUBSYS directory.

## EXAMPLES

```
$ crsubs newsubs          Create the subsystem "newsubs".
```

**CRTY (CREATE\_TYPE) -- Create a new type.**

## FORMAT

**CRTY type\_name [options]**

The CRTY command creates new types. It creates an identifier for the new type, and associates it with the supplied type name. New types are used to identify a new kind of manager for Streams.

## ARGUMENTS

**type\_name**  
(required)

Specify the name to assign to the created type.

## OPTIONS

**-N node\_spec**

Specify the node on which the type is to be created. See the section on node specifications in Chapter 3 for more information. You may also specify the entry directory of a volume mounted for software installation, as shown in the example below. If this option is omitted, the type is created on the current node.

**-L**

List the type name/type identifier pair that is created.

**-U high.low**

Create the type with the specified unique identifier (UID). Give the high and low addresses for the UID as indicated. *Use this option only for system debugging purposes.* Misuse of this option may cause programs to behave incorrectly.

**-B[INARY] pathname**

Create the type from the specified object module (which was created by CRTYOBJ). This allows you to add a new type to a system from an object module which can be shipped on media like floppies, magnetic tapes, etc.

## EXAMPLES

```
$ crty example_type -l
"example_type" 24BF9F41.100001FB created.
```

```
$ crty example_type -n //test_vol -l
"example_type" 24BFA6F8.200001FB created on volume //test_vol.
```

In the following example, the disk has been mounted for software installation. The disk's top level directory (catalogued as '/mount\_disk' by the MTVOL command) must contain a "sys" directory. If it does not, you will get a "type manager directory not found" error.

```
$ mtvol w /mount_disk
$ crty example_type -n /mount_disk -l
"example_type" 24BFB71E.200001FB created on volume //my_node/mount_disk.
```

CRTYOBJ (CREATE\_TYPE\_OBJECT)

CRTYOBJ (CREATE\_TYPE\_OBJECT) -- Create a type object module for binding.

## FORMAT

CRTYOBJ **type\_name** [**variable\_name**] [**options**]

CRTYOBJ creates an object module which contains a global symbol with the type UID. This module is bound with type managers. The variable is passed into calls to TRAIT\_\$MGR\_DCL to declare support for the specified type.

## ARGUMENTS

**type\_name**

(required)

Specify the name of the type for which an object module is to be created.

**variable\_name**

(optional)

Specify the variable name for the type uid.

Default if omitted: name the variable "type\_name\_\$UID".

## OPTIONS

**-B bin\_name**

Specify the output binary file name. The default is "type\_name.BIN".

**-SECT section\_name**

Specify the section name for the data area in which the variable is declared. The default section name is "DATA\$".

**-U high.low**

Specify the type UID explicitly with the high and low addresses in the positions indicated. *Use this option only for system debugging purposes.*

**-PRE\_SR9.5**

Generate an object module in the pre-SR9.5 format. If this option is omitted, the generated object module is in the SR9.5 format.

## EXAMPLES

```
$ crtyobj example_type example_$uid
$ bind -b example_mgr example_main.bin example_calls.bin example_type.bin
```

**CRUCR (CREATE\_UCR) -- Create a User Change Request form.**

## FORMAT

### CRUCR

We appreciate feedback from users of our systems. To make the feedback process as easy as possible, we provide User Change Request (UCR) forms for you to record comments and suggestions.

CRUCR enables you to complete user change request forms on line. CRUCR assigns each form a unique number and stores the completed form in /SYS/UCR. /SYS/UCR may be a link.

The first time you execute CRUCR on a node, the command prompts you for information about your site and creates the object file /SYS/UCR/DMPROCxxxx, where xxxx is your node ID. This file contains the default UCR form that is used from then on. You may edit this template to add information to subsequent UCR forms.

Submit UCRs to us on floppy disks or as line-printer listings, using a separate UCR for each distinct problem that you report. If you are sending in a program, please use a floppy disk. Mail the material to:

Apollo Computer Inc.  
330 Billerica Road  
Chelmsford, MA. 01824

Attn: UCR Administrator, Technical Operations

You may also send UCRs directly to Apollo Customer Services via the UUCP network.

The network address is: ucr\_admin@apollo.uucp

Recommended paths to Apollo are via uw-beaver and decvax!wanginst.

When you send UCRs via UUCP, please supply the output from the standard CRUCR command. Customer Services will acknowledge all received UCRs. Therefore, do not assume your UCR has been received unless you receive a reply. Security-conscious sites should not send confidential material. Also, you should send voluminous submissions by magnetic media.

CRUCR does not require any arguments or options.

CSR (COMMAND\_SEARCH\_RULES)

CSR (COMMAND\_SEARCH\_RULES) -- Set or display command search rules.

## FORMAT

CSR [directory ...] [-A dir\_name]

Command search rules determine which directories the Shell examines to find commands. CSR lets you display or change this list. If a new Shell is invoked *inside the current process*, or a new Shell script run, the subordinate Shell inherits the search rules of the parent Shell. Note that this does *not* apply to "shells" created by the <SHELL> key, since that key actually creates a new, separate process. Its Shell receives the default search rules described below.

By default, the Shell looks for commands in this order:

1. Your working directory ("."), or the directory specified by the command's pathname.
2. Your personal command directory, ~COM (the COM subdirectory of your naming directory).
3. The system command directory, /COM.

Refer to *The DOMAIN System User's Guide* for a detailed discussion of command search rules.

Specifying CSR without arguments or options displays the current command search rules.

## ARGUMENTS

**directory**  
(optional)

Specify new command search sequence. Multiple directory pathnames are permitted; separate names with blanks. The Shell will search the directories in the order that you specify.

Default if omitted: display current search rules unless -A is specified.

## OPTIONS

**-A dir\_name**

Append the specified directory name(s) to the existing command search sequence. This allows you to add a new directory to the end of the list without retyping the entire list. Multiple directory pathnames are permitted; separate names with blanks.

## EXAMPLES

1. \$ csr  
    ~com /com

Display current search rules.

## CSR (COMMAND\_SEARCH\_RULES)

2. `$ csr . ~com //us/myproj/com /com`

Set new search sequence by adding an additional command directory.

3. `$ csr -a ~com/special_commands`

Append the directory ~com/special\_commands to the current list of directory names.

CTNODE (CATALOG \_NODE)

CTNODE (CATALOG \_NODE) -- **Catalog a node in the network.**

## FORMAT

**CTNODE** [**node\_name** [**net.**]node\_id ...] [**options**]

CTNODE informs the local node that a remote node exists, thereby enabling network file access to the remote node. The command catalogs the node\_name in the local copy of the network root directory as the entry directory for the remote node. In other words, CTNODE adds the directory //node\_name to your copy of the network root directory. For information on deleting a node\_name entry, see UCTNODE (UNCATALOG \_NODE).

We assign a node ID to every node during the manufacturing process. To find out the node ID of a node, type the following command at its keyboard:

```
$ lnode -me
```

At SR9.0, CTNODE supports the ability to merge information from another node's network root into your own, or any other node's network root. The merge options (-MD and -MS) add the entry for a node to the target provided the entry does not already exist and the source has exactly one entry for that node. In the case of one source and one target entry for a node which match, those entries are assumed to be correct. All other cases are considered to be ambiguous and the "confusion resolution protocol" is invoked.

This "confusion resolution protocol" first attempts to verify the correct entry name with the node itself. If the node is available, then the reply from the node is cataloged *regardless of whether -MD or -MS is used*. This is because an answer from the node itself is assumed to be the truth.

If the node is unavailable to resolve an ambiguity, then the entry containing the most recent UID (latest time stamp portion of the UID), is used. In this case, existing entries in the target directory are only updated if the -MS option is used.

## ARGUMENTS

**node\_name**

(optional)

Specify the name of the node you wish to catalog. If the '[net.]node\_id' argument is specified, then 'node\_name' is required.

Default if omitted: must use -N, -UPDATE, or -FROM

[net.]node\_id

(optional)

Specify the hexadecimal ID (and optional network ID) of the node you wish to catalog. The node must be connected to the network when this command is executed. If the 'node\_name' argument is specified, then 'node\_id' or 'net.node\_id' is required.

Default if omitted: must use -N, -UPDATE, or -FROM

Multiple name/ID pairs are permitted.

## OPTIONS

If neither **-N**, **-UPDATE**, or **-FROM** is specified, then the `'node_name'` and `'[net.]node_id'` arguments are required. The **-N**, **-UPDATE**, and merge options work only for remote nodes running AEGIS SR5.0 or later. The `'[net.]node_id'` forms work only when both the local and remote nodes run AEGIS SR9.0 or later.

**-ROOT**                   Catalog `'node_name'` as the entry directory name for `'node_id'` in both the master network root directory and the local copy of the network root directory. This option is valid only if the `'node_name'` and `'node_id'` arguments are specified. This option is *not* valid if the **-N** option is specified.

**-N [net.]node\_id...**   Copy the entry directory name from the network root directory of the specified remote node, to the network root directory of the local node. You do not need to know the entry directory name. However, you must specify the `node_id` or the `net.node_id` of the remote node. Multiple `node_id`'s and `net.node_id`'s may be specified. Use this option instead of the `'node_name'/'[net.]node_id'` argument pair. This option is *not* valid if the **-R** option is specified.

**-UPDATE**                Obtain a list of nodes currently responding to a network inquiry and perform the same operation as **"-N"** for each node. Names are replaced with the most current version, if they already exist in your local copy of the network root directory, and new names are added.

**-FROM //node ...**       Look in the specified list of network root directories for the names to add to the target network root, or use this network root as the source for names to merge into the target network root. Wildcards may be used to specify source node names. The **-FROM** option is not supported in a DOMAIN internet environment.

**-MD**                    Used with **-FROM**. Merges all names in the source network root into the target network root. Preference is given to existing names in the target if there are unresolved conflicts (see the discussion of "confusion resolution protocol" above).

**-MS**                    Same as **-MD**, except that preference is given to entries in the source network root when there are unresolved conflicts (see the discussion of "confusion resolution protocol" above).

**-ON //node ...**        Catalog names in the network root of the specified nodes instead of the local network root. Wildcards may be used to specify target node names. The **-ON** option is not supported in a DOMAIN internet environment.

**-R**                     Replace cataloged names if they already exist. An error occurs if you do not specify this option and try to add a `node_name`

## CTNODE (CATALOG\_NODE)

that has already been cataloged (unless you are using -UPDATE).

- L List node names as they are cataloged.
- IDUPL Ignore entry (suppress error messages) if name already exists in the target.
- LC List invocations and resolutions of the "confusion resolution protocol".

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### EXAMPLES

1. `$ ctnode os 21` Add the node whose ID is 21 and whose entry directory name is "os" to your node's catalog.
2. `$ ctnode -update` Bring your node's catalog up to date with any new nodes on the network.
3. `$ ctnode os eve -from //master` Copy names "os" and "eve" from the network root on //master.
4. `$ ctnode os 21 -on //a?*` Add node ID 21 with the name "os" to the network root of all nodes whose names begin with "A".
5. `$ ctnode -md -from //os` Merge network root of OS into local network root, resolving conflicts.

**CTOB (CATALOG\_OBJECT) -- Catalog an object.**

#### FORMAT

**CTOB pathname uid\_hi uid\_low**

CTOB assigns a pathname to an object that has a known unique identifier (UID). CTOB catalogs the pathname and associated UID in the naming tree. This command is primarily for system-level debugging.

#### ARGUMENTS

**pathname**  
(required)

Specify assigned pathname.

**uid\_hi**  
(required)

Specify the high portion of the UID as a 32-bit hexadecimal number.

**uid\_low**  
(required)

Specify the low portion of the UID as a 32-bit hexadecimal number.

#### OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

#### EXAMPLES

```
$ ctob lastfile 10A0BAAD 60000102
```

CVT\_REC\_UASC (CONVERT\_RECORD\_UASC)

CVT\_REC\_UASC (CONVERT\_RECORD\_UASC) -- Convert file types.

## FORMAT

CVT\_REC\_UASC **source\_pathname** [**target\_pathname**] -OT **type** [**options**]

CVT\_REC\_UASC converts files from type "rec", "hdru", or "uasc" to files of type "rec", "hdru", or "uasc". It functions on nodes running software release 4.1 and later.

## ARGUMENTS

**source\_pathname**

(required)

Specify the file to be converted.

**target\_pathname**

(optional)

Specify file to be created. An error occurs if this file already exists (see -R below). The **target\_pathname** may be derived. If target is a directory, the source file is converted and placed in that directory.

Default if omitted: the converted file becomes 'source\_pathname' and the original file is renamed 'source\_pathname.CBAK'.

**-OT type**

(required)

Specify type of file to be created ('target\_pathname'). Choose one of the following for 'type': "rec", "hdru", or "uasc."

Wildcards in pathnames associated with this command are permitted.

## OPTIONS

**-R**

Replace 'target\_pathname' if it already exists.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

1. \$ `ld -a` List current files in specified directory and their types.

Directory `"/larry/cvt_rec_uasc_examples"`:

sys	type	blocks	current				
type	uid	used	length	attr	rights	name	
file	rec	1	42	P	pndwrx	a	
file	rec	1	42	P	pndwrx	b	
file	rec	1	44	P	pndwrx	c	

3 entries, 3 blocks used.

2. \$ `cvt_rec_uasc ?* -ot uasc -nq` Convert all files to type uasc;  
 \$ `ld -a` suppress wildcard queries.

Directory `"/larry/cvt_rec_uasc_examples"`:

sys	type	blocks	current				
type	uid	used	length	attr	rights	name	
file	uasc	1	37	P	pndwrx	a	
file	rec	1	42	P	pndwrx	a.cbak	
file	uasc	1	38	P	pndwrx	b	
file	rec	1	42	P	pndwrx	b.cbak	
file	uasc	1	40	P	pndwrx	c	
file	rec	1	44	P	pndwrx	c.cbak	

6 entries, 6 blocks used.

3. \$ `cvt_rec_uasc [a-c] =.x -ot rec -nq` Convert files named "a" "b"  
 \$ `ld -a` and "c" to type rec and write  
 them to "a.x" "b.x" and "c.x"

Directory `"/larry/cvt_rec_uasc_examples"`:

sys	type	blocks	current				
type	uid	used	length	attr	rights	name	
file	uasc	1	37	P	pndwrx	a	
file	rec	1	42	P	pndwrx	a.cbak	
file	rec	1	42	P	pndwrx	a.x	
file	uasc	1	38	P	pndwrx	b	
file	rec	1	42	P	pndwrx	b.cbak	
file	rec	1	42	P	pndwrx	b.x	
file	uasc	1	40	P	pndwrx	c	
file	rec	1	44	P	pndwrx	c.cbak	
file	rec	1	44	P	pndwrx	c.x	

9 entries, 9 blocks used.

\$

## DATE

**DATE --** Display the current date and time.

## FORMAT

**DATE [options]**

DATE prints the current system date and time. It requires no arguments or options. If no options are specified, the date is displayed as shown in Example 1 below.

The hardware date and time may be set with the Shell command CALENDAR.

## OPTIONS

- Y** Display year as YYYY.
- MD** Display month and day as MM/DD.
- T** Display time in 24-hour format (HH:MM:SS).
- D** Display year, month, and day.

## EXAMPLES

1. `$ date`  
Tuesday, June 9, 1981 4:20:15 pm (EDT)
2. `$ date -t`  
15:36:14
3. `$ date -d`  
1983/08/08

**DCALC (DESK\_CALCULATOR) -- Evaluate logical and arithmetic expressions.**

## FORMAT

**DCALC [-H] [pathname...]**

DCALC mimics the features of a desk calculator, evaluating both logical and arithmetic expressions.

## ARGUMENTS

**pathname**  
**(optional)**

Specify input file containing expressions to be evaluated, one expression per line.

Default if omitted: read standard input; stop with CTRL/Z

## OPTIONS

If no options are specified, all operations are decimal-based.

**-H** Specify hexadecimal operations.

### *Expressions*

Input expressions can be simple arithmetic expressions or variable assignment expressions. DCALC writes the value of each evaluated expression on standard output. Variables hold temporary values, which DCALC does not automatically write.

Expressions may include any of the operators listed below (in order of precedence):

1.   + -       unary plus and negation operators. These may only appear at the start of an expression or within parentheses.
2.   << >>     logical left and right shift
3.   \*\*        exponentiation
4.   \* / %     multiply, divide, modulo (remainder)
5.   + -       add, subtract
6.   ==        equal to
- !=       not equal to
- >        greater than
- >=       greater than or equal to

## DCALC (DESK\_CALCULATOR)

	<	less than
	<=	less than or equal to
7.	!	unary logical not
8.		logical or
	&	logical and
	^	logical xor

Relational operators return the value 1 for true and 0 for false. DCALC performs operations in double precision floating point, except for logical operators listed as items 2 and 8 above, which use 32-bit integers.

### *Variables*

Expressions may include previously declared variables. Use this format to declare a variable:

```
name = expression
```

- A variable name must begin with a letter and may consist of any combination of letters and digits.
- DCALC does not automatically print replacement expressions, because they usually contain temporary values.

### *Radix Control*

You can change the default base for input or output using `ibase` (input base) and `obase` (output base) statements. For example,

```
ibase = 2
```

```
obase = 16
```

causes DCALC to interpret input in binary and print results in hexadecimal.

To set an individual number's radix, precede it with the desired radix and a pound sign. For example,

```
16#100
```

specifies the hexadecimal number 100 (equals 256 in decimal).

## EXAMPLES

Your input:	DCALC output:
1. 10 + (-64 / 2**4)	6
2. temp = 2#101 temp == 5	1 (true)
3. ibase = 16 obase = 2 11 + 28 1a + 0f	111001 101001

(Note that when you type a hexadecimal number that begins with a letter, you must precede it with a zero.)

4. ibase = 16 numa = 100 numb = 100 numa + numb	512
--	-----

## DEBUG

**DEBUG -- Invoke the Language Level Debugger.**

### FORMAT

**DEBUG [options] {-PROC process\_name | target [args...]}**

The Language Level Debugger (DEBUG) lets you debug programs written in Pascal, FORTRAN, or C. The *DOMAIN Language Level Debugger Reference* manual details the debugger. After you've invoked it, you can enter DEBUG's 'HELP' command to read other help files that explain how to operate the debugger.

#### *Preparing a File for Debugging*

In order to use DEBUG to debug a program, you must have compiled your source code with the correct compiler option. Each compiler supports four command options (-DB, -DBA, -DBS, or -NDB) that affect DEBUG's access to a program.

-DB, the default, gives you limited access to the debugger. -DBA and -DBS both give you full access to the debugger. With -DBA, the compiler removes any optimizations which might interfere with debugging. With -DBS, the compiler allows any optimizations specified by the -OPT option. If you compile with -NDB, you cannot debug the file.

#### *Invoking the Debugger*

To invoke DEBUG, enter a command having one of the following two formats:

```
$ DEBUG debug_options target [args...]
```

or

```
$ DEBUG debug_options -PROC process_name
```

The necessary arguments and options are described below. By default, the window from which you invoke DEBUG is divided into three windowpanes. One windowpane will contain all I/O for the program you are debugging. Another windowpane will contain all the DEBUG commands you enter and all the responses from DEBUG. A third windowpane will display the source code of the program you are debugging.

### ARGUMENTS

One of the following two arguments is required on the command line.

**target [args...]**

**(optional)**

Specify the pathname of the file containing the program you wish to debug, plus any arguments which that program may require.

**-PROC process\_name**

**(optional)**

Perform explicit cross-process debugging. This primes DEBUG to watch for target invocation in an already-existing process (specified by 'process\_name'). After you invoke DEBUG with this option, it will watch the given process for the invocation of a program. When the invocation of the target program occurs,

DEBUG will wake up and take control of the target program, and give you the regular DEBUG access to it. You cannot debug an already running program, or a process which is already the target of another debugger.

This feature is especially useful when the normal invocation of DEBUG and a target perturbs the environment enough to make a bug disappear. In this case, creating a new process for DEBUG and directing it to watch the old process will help ensure that the target runs in the same environment as when it runs alone. Cross-process debugging is also helpful for programs which perform graphics or in some way alter or control the window(s) of the process in which they run, thus making normal, within-process debugging impossible.

## OPTIONS

Note that all DEBUG options MUST PRECEDE the pathname of the target program.

Default options are indicated by "(D)."

**-NC** Prevent DEBUG from copying the target object file. Instead, DEBUG maps the object file so that you can write breakpoints directly into the object file.

**-R[EAD] pathname** Invoke a DEBUG command file with the specified pathname. This option may appear only once on the command line.

**-SET arg\_string** Set debug variable prior to invoking the target program. 'arg\_string' is the body of a valid Set command. The string must be quoted if it contains spaces, so that the command line parser sees it as one argument. See Example 2 below. If you submit a -SET option which does not have an assignment operator (=, :=) in 'arg\_string', the Set command will be interactive as it is during normal DEBUG operation.

**-WPn** Specify size of DEBUG windowpane from 10% to 90%. 'n' may equal 10, 20, 30, 40, 50, 60, 70, 80, or 90. Default if omitted: DEBUG creates a windowpane in the top 50% of the window.

**-NWP** Do not create DEBUG windowpanes. Instead, DEBUG will perform input and output operations using the error input and error output streams in your transcript pad.

**-SRC**

**-SRC\_T**

**-SRC\_R (D)** These options cause DEBUG to display the source file(s) which were used to make the target program being debugged. The -SRC option lets DEBUG choose where the source will be displayed. The -SRC\_T option makes DEBUG put the source at the top of the window; the -SRC\_R option makes DEBUG put the source on the right-hand side of the window.

## DEBUG

- NSRC** Suppress creation of a source-display windowpane.
- SDIR pathname** This option provides alternative directory pathnames for finding the source file(s), when one of the **-SRC** options is used. It may be given any number of times on the command line. The working directory is always checked and hence need not be specified.
- GLOB** Enable DEBUG to enter routines in global address space.
- SMAP** Print a brief section map of the target program loading operation.

### *DEBUG Startup Files*

When you invoke DEBUG it looks in your login home directory for a file named "user\_data/startup\_debug". If it finds the file DEBUG processes its contents as a sequence of DEBUG commands. DEBUG then looks in the working directory for a file named "startup\_debug" and similarly processes it. No error occurs if one or both files are not found. Startup file processing precedes processing of a file given in a **-READ** option.

### *Saving a DEBUG Session*

To save your dialog with DEBUG, use the Display Manager's command "PN" to name the debugger's transcript pad anytime before you issue the debugger's quit command. See the PN command description for more information.

## EXAMPLES

1. `$ debug my_prog` Tells the debugger to debug the file named my\_prog.
2. `$ debug -read s1 my_prog` Invokes the debugger for a debugging session with file my\_prog, and starts the session by executing the commands stored in file s1.
3. `$ debug -src -set "'max_array_dim = 8" my_prog` Sets the 'MAX\_ARRAY\_DIM variable as you invoke DEBUG; allows DEBUG to select the most appropriate type of source display.

**DLDUPL (DELETE\_DUPLICATE\_LINES) -- Strip repeated lines from a file.**

## FORMAT

**DLDUPL [-C] [pathname ...]**

DLDUPL reads the input file(s), comparing adjacent lines. Second and succeeding copies of repeated lines are removed; the remaining lines are written to standard output.

## ARGUMENTS

**pathname**  
(optional)

Specify input file. Multiple file names permitted; separate names with blanks.

Default if omitted: read standard input

## OPTIONS

**-C**

Write number of occurrences of each line to standard output.

## EXAMPLES

Suppose you have two dictionary files. To create one dictionary file containing the words from both, use:

```
$ srf -m dict1 dict2 | dldupl >dict.new
```

This merges the words from the two files (SRF -M), then deletes any duplicate words and saves the result in the new dictionary.

**DLF (DELETE\_FILE)**

**DLF (DELETE\_FILE) -- Delete one or more files.**

## FORMAT

**DLF [pathname...] [options]**

DLF deletes the file(s) specified. To delete objects other than files, see DLL (DELETE\_LINK) and DLT (DELETE\_TREE).

## ARGUMENTS

**pathname**  
**(optional)**

Specify file to be deleted. Multiple names and wildcarding are permitted; separate names with blanks.

Default if omitted: read names from standard input

## OPTIONS

- F** Force file deletion if you have owner rights, even if you don't have delete rights.
- L** List names of deleted files.
- DU** Delete when unlocked. If the object to be deleted is locked when DLF is invoked, the delete operation will be performed when the object is unlocked.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ dlf mary.bak -L
(file) "mary.bak" deleted.
```

**DLL (DELETE\_LINK) -- Delete a link.**

### FORMAT

**DLL pathname ... [options]**

DLL deletes a link. After execution of this command, the link is no longer available for use.

### ARGUMENTS

**pathname**  
(required)

Specify pathname of the link to be deleted. Multiple pathnames and wildcarding are permitted; separate names with blanks.

### OPTIONS

**-L** List name(s) of link(s) as deleted.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### EXAMPLES

```
$ dll bugs  
$
```

Delete the link "bugs" from the current working directory.

DLT (DELETE\_TREE)

DLT (DELETE\_TREE) -- Delete a tree.

## FORMAT

**DLT** *pathname* ... [*options*]

DLT deletes the directory named by the *pathname*, and all its descendants in the naming tree.

## ARGUMENTS

**pathname**  
(required)

Specify directory or link to be deleted. If "pathname" is a directory, DLT deletes the directory and all subordinate objects (subdirectories, files, and links). If a link, DLT deletes the link name, but has no effect on the files and directories named by the link. Multiple pathnames and wildcarding are permitted.

## OPTIONS

- L** List files, links, and directories as they are deleted.
- LD** List directories as they are deleted.
- LF** List files as they are deleted.
- LL** List links as they are deleted.
- F** Force object deletion if you have owner rights, even if you don't have delete rights.
- DU** Delete when unlocked. If the object to be deleted is locked when DLT is invoked, the delete operation will be performed when the object is unlocked.
- PR *pathname*** Preserve specified pathnames.

-LD, -LF, and -LL may be combined to create the type of listing you desire.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ dlt april_backup may_backup      Delete the two directory trees
                                     specified.
```

**DLTY (DELETE\_TYPE) -- Delete a type.**

## FORMAT

**DLTY type\_name [options]**

DLTY deletes a type and any installed type manager.

## ARGUMENTS

**type\_name**  
(required)                      Specify the name of the type to be deleted.

## OPTIONS

**-N node\_spec**                      Specify the node on which the type is to be deleted. See the section on node specifications in Chapter 3 for more information. You may also specify the entry directory of a volume mounted for software updates, as shown in the example below. If this option is omitted, the type is deleted on the current node.

**-L**                                      List the type name/type identifier pair that is deleted.

## EXAMPLES

```
$ dltypes example_type -l
"example_type" 24BF9F41.100001FB deleted.
```

```
$ dltypes example_type -n //test_vol -l
"example_type" 24BFA6F8.200001FB deleted from volume //test_vol.
```

In the following example, the disk has been mounted for software updates. The disk's top level directory (catalogued as '/mount\_disk' by the MTVOL command) must contain a "sys" directory. If it does not, you will get a "types file not found" error.

```
$ mtvol w /mount_disk
$ dltypes example_type -n /mount_disk -l
"example_type" 24BFB71E.200001FB deleted from volume //my_node/mount_disk.
```

**DLVAR (DELETE\_VARIABLE)**

**DLVAR (DELETE\_VARIABLE) -- Deletes all of the specified variables.**

### **FORMAT**

**DLVAR var\_name ...**

The DLVAR command deletes the variable(s) specified. If a variable had another value at a higher level of invocation, the variable is restored to that value.

### **ARGUMENTS**

**var\_name ...**

**(required)**

Specify the variable name to be deleted. Multiple names are permitted, separated by blanks.

**DMTVOL (DISMOUNT\_VOLUME) -- Dismount a logical volume.**

## FORMAT

**DMTVOL device[unit] [log\_vol\_number] [pathname] [options]**

DMTVOL dismounts a logical volume that was previously mounted with the MOUNT\_VOLUME (MOUNT\_VOLUME) command. After the volume has been dismounted, it is unavailable for further access.

## ARGUMENTS

**device**  
(required)                      Specify the type of disk on which the volume resides: "W" for a Winchester disk, "S" for a storage module, or "F" for a floppy disk.

**unit**  
(optional)                      Specify a unit number (0 or 1 only) for the device, if necessary. For example, "S1" denotes storage module unit 1.  
Default if omitted: 0 (zero)

**log\_vol\_number**  
(optional)                      Specify the number of the logical volume to be dismounted.  
Default if omitted: 1

**pathname**  
(optional)                      Specify the entry directory of the logical volume. If you include this argument, DMTVOL dismounts the volume and uncatalogs its entry directory. If you omit it, DMTVOL dismounts the logical volume, but retains its name in the naming tree.  
Default if omitted: see above

## OPTIONS

**-FU**                              Forcibly unlock any locked objects, then dismount the volume. If you omit this option, the dismount fails if the volume contains any locked objects.

**-NW**                              No\_write -- Prevents DMTVOL from trying to write to the disk during the dismount. Normally, writing to the disk saves current information. However, if the disk was removed prior to dismount, this option should be used.

## DMTVOL (DISMOUNT\_VOLUME)

### EXAMPLES

1. \$ dmtvol s 2  
\$

Dismount storage module unit zero,  
logical volume 2, and leave its name in  
the naming tree.

2. \$ dmtvol f /floppy  
\$

Dismount floppy unit zero, logical volume 1,  
and delete its name from the naming tree.

**DSPST (DISPLAY\_PROCESS\_STATUS) -- Display process status graphically.**

**FORMAT**

**DSPST [options]**

DSPST displays process statistics in a graphical, bar-chart fashion within the current process window. The chart is updated periodically (see -R below). The default action of this command is to display the brief OS process list, all user processes and all I/O information in a font size automatically selected based on window size.

While DSPST is running, the following keys are interpreted as follows:

All keyboards:

^T	Move to top
^B	Move to bottom
RETURN	Exit
^N	Exit
^Y	Exit and save current image

880 keyboard only:

Boxed up arrow	Scroll forward 1/2 window
Boxed down arrow	Scroll backward 1/2 window
F2	Scroll backward 1 line
F3	Scroll forward 1 line

Low-profile keyboard only:

Boxed up arrow	Scroll backward 1/2 window
Boxed down arrow	Scroll forward 1/2 window
Shifted up arrow	Scroll backward 1 line
Shifted down arrow	Scroll forward 1 line
EXIT or ABORT	Exit
SAVE	Exit and save current image

**OPTIONS**

Default options are indicated by "(D)."

- R n** Specify that the display should be repeatedly updated every 'n' seconds. If this option is omitted, the display is updated every 4 seconds.
- P** Show process information.
- L1** Show OS and user process information.
- OS (D)** Show brief OS and full user process information.
- M** Show missing CPU time in addition.

## DSPST (DISPLAY\_PROCESS\_STATUS)

- IO** (D) Show I/O statistics.
- A** Show all information (same as -L1 -IO -M).
- N node\_spec** Specify remote node whose process statistics are to be listed. See the section on node specifications in Chapter 3 for more information.
- LARGE** (D) Force use of large font for display.
- SMALL** Force use of small font for display.

### EXAMPLES

- `$ dspst` Display OS, user process, and I/O status.
- `$ dspst -n //fred -large` Display OS, user process, and I/O status for the node named //fred using the large font.

**ED (EDIT) -- Invoke line editor.**

## FORMAT

**ED [-N] [pathname]**

ED invokes the line editor. Input text and editing commands are read from standard input. While you may use ED to create text files interactively, it is better suited for use in programs and scripts. Use the <EDIT> key or the DM command, CE, to create and edit files interactively.

**NOTE:** There is a homonymous DM command: ED -- Delete character preceding cursor. See the ED command description in the DM chapter for details.

## ARGUMENTS

**pathname**  
**(optional)**

Specify file to be edited. ED reads the file into a buffer for editing and remembers its name for future use. ED operates on the buffer copy; changes made there have no effect on the original file until you issue a W (write) command from within ED. Files are limited to 6400 lines.

If the 'pathname' argument is omitted, the edit buffer is empty and no file name is remembered for future use. You will have to specify an explicit file name when you exit the editor.

Default if omitted: see above

## OPTIONS

**-N** Suppress the printing of line counts by the E (edit), R (read), and W (write) commands.

## SUMMARY OF ED COMMANDS

Commands to ED have a consistent format: zero, one, or two line addresses followed by a single-character command, with optional parameters following the command. The general format is:

[line.] [line] command parameters

The [line] specifies a line number or address in the current edit buffer. There is usually a useful default for each command (normally the current line) so that you don't need to specify an address explicitly.

Refer to the ED description in the *DOMAIN System Utilities* manual for detailed explanations of individual ED commands.

## ED (EDIT)

### Addresses:

17      a decimal number  
      .      the current line  
      \$      the last line of the file  
      /pat/    search forward for line containing pat  
      \pat\    search backward for line containing pat  
      line+n   n lines forward from line  
      line-n   n lines backward from line

### Defaults:

(.)      use current line  
(.+1)    use the next line  
(...)    use current line for both line numbers  
(1,\$)    use all lines

### Commands:

(.)      A            Append text after line (text follows)  
(..n)    Bn            Browse over the next n lines (default n is 22).  
          If n is negative, print last n lines before  
          current line. If 'B.' is specified, print n  
          lines with current line in center of screen.  
(...)    C            Change text (text follows)  
(...)    D            Delete text  
          E file      Discard current text, enter file, remember  
                      filename  
          F            Print filename  
          F file      Remember filename  
(.)      I            Insert text before line (text follows)  
(...)    Kline        Copy text to new line after specified line  
(...)    Mline        Move text to line after specified line  
(...)    P            Print text (can be appended to other commands)  
          Q            Quit  
(.)      R [file]     Read file, appending after current line  
(...)    S/pat/new/GP    Substitute new for leftmost pat (G implies all  
          occurrences)  
(1,\$)    W [file]     Write file, leave current text unaltered (if  
          no file is specified, write to current filename)  
(.)      =[P]        Print line number, current line  
(.+1)    <CR>        Print next line  
(1,\$)    G/pat/command   Execute command on lines containing pat  
          (except A, C, I, Q commands)  
(1,\$)    X/pat/command   Execute command on lines not containing pat  
          (except A, C, I, Q commands)  
          # ...      Comment  
          \$n        Read or write temporary buffer, "n".

The error message "?" is printed whenever a command fails or is not understood.

### LIMITATIONS

- Files being edited can contain up to 6400 lines.
- When a global search and substitute combination fails, the entire global search stops.
- Problems sometimes occur when removing or inserting NEWLINE characters (via @n), especially in global commands.

**EDACCT (EDIT\_ACCT\_FILE) -- Create, edit, or list accounts.**

## FORMAT

**EDACCT [options]**

EDACCT is used to define accounts in the network registry. Valid person, project, and organization names must have been previously defined with EDPPO.

While all of the EDACCT options are described below, it is unlikely that you will be able to manipulate the network registry unless you are the network administrator for your network. The registry is protected by ACL restrictions. However, you will be able to list registry entries.

For complete information on the use of network administration commands such as EDACCT, see *Administering Your DOMAIN System*.

## OPTIONS

At least one of the following options must be specified; however, you may only include one -A, -D, -C, or -L option per command line. If the command line does NOT include -A, -D, -C, or -L, EDACCT enters an interactive editing session and accepts commands from standard input. See the "EDACCT COMMANDS" section below.

**-R pathname** Specify pathname of registry you want to use. You should only use this option with -LOC (described below) to manipulate a remote node's local registry. If you want to manipulate the master registry, omit this option and let EDACCT use the network registry file copy ('/registry/registry') on the current node to locate the master registry.

**-A pers proj org homedir password** Add a new account with the person, project, organization, log in home directory, and password indicated. You must specify all fields; you can use "None" to specify proj and/or org. A space between two quotation marks specifies a blank password. This option is not valid for local registries.

**-D pers proj org** Delete the account with the person, project, and organization names indicated. You must specify all fields. This option is not valid for local registries.

**-C pers proj org [-p passw] [-h homedir]** Change the password and/or the login home directory for the account named. A space between two quotes may be used to specify a blank password. This option is not valid for local registries.

**-L [pers [proj [org]]]** List specified entries. All entries for a particular category are listed if you omit that portion of the account specification. You

## EDACCT (EDIT\_ACCT\_FILE)

may also use "%" in any of the fields to match all entries in that field. If you indicate no accounts by name, all accounts are listed.

- LOC** Use the registry '/registry/local\_registry' on the node given by the -R option or this node. You may specify only the node name (//node) for "pathname" on -R in this case. If you specify -LOC, the only valid operation is to list entries (-L).
- NP** Suppress prompts for interactive editing.

### EXAMPLES

```
$ edacct -loc -l          List entries in the local account file.
                          Note that two have expired.

paul   mfg       none    30 83/03/12.12:16   exp:83/03/12 //pj/paul
joe    none     r_d     1FB 83/03/15.08:28   exp:83/03/15 //bye/joe
csa    none     none    30 83/03/15.08:32   exp:83/03/15 //my/csa
rjm    mktg     pay     B8 83/03/15.09:50   exp:83/03/15 //slash/rjm
zoo    cage     feed    1FB 83/03/14.20:53   exp:83/03/14 *INVALID* //me
flip   none     wilt    124 83/03/09.18:46   exp:83/03/09 //go/flip
user   none     none    30 83/03/14.19:22   exp:83/03/14 *INVALID* /
```

### EDACCT COMMANDS

The following interactive editing commands may be entered from standard input if you do not include -A, -D, -C, or -L on the EDACCT command line. In all cases, periods may be used instead of blanks to separate person, project, and organization identifiers.

a [pers [proj [org [homedir [password]]]]]

Add a new account. The account is added AFTER the current account (last one listed) unless you have positioned to the top (t). You will be prompted for any fields omitted. Press <RETURN> to assign the default, if appropriate. A space between two quotes may be used to specify a blank password.

d [pers [proj [org]]]

Delete one or more accounts. If no fields are given, only the current entry (last one listed) is deleted. % may be specified for wildcard match, in which case all matching entries (from the top) are deleted. If proj and/or org are omitted, % is assumed for them. Verification of each deletion is requested. The last deleted entry is saved and may be moved to a different place using 'i'.

i Insert the last deleted entry at the current location (after the current entry or at the top).

t Move to the top of the file, before the first entry.

b Move to the bottom of the file, at the last entry.

n [[-]number]

Move up (-) or down "number" entries. For example, "n 2" moves down two entries, while "n -4" moves up four entries. If "number" is omitted, the default is 1.

c [pers [proj [org]]]

Change one or more entries. If no arguments are specified, only the current entry is changed. % specifies wildcard match. If proj and/or org are omitted, % is assumed for them. If arguments are given, the search begins at the top of the file. Wildcarded fields are not eligible for changing. You will be prompted for each new field in order. Press <RETURN> to keep a field unchanged.

l [pers [proj [org]]]

List the current entry (if no arguments are specified) or the first matching entry, starting at the top. % may be used to specify wildcard match.

la [pers [proj [org]]]

List all matching entries, starting at the top. % may be used to specify wildcard match.

ln [pers [proj [org]]]

List the next matching entry, starting at the current entry. If no arguments are given, the previous name specification is used.

lc List all the changed or new entries.

ll List the locality of the current entry, displaying the preceding and succeeding five entries.

wr Update the file with all changes and exit.

q Quit without updating.

h [comm] Help -- list briefly all the available commands or a particular command in detail.

EDACL (EDIT\_ACCESS\_CONTROL\_LIST)

EDACL (EDIT\_ACCESS\_CONTROL\_LIST) -- Edit or list an ACL.

## FORMAT

EDACL [commands] [options] pathname...

Every directory and file has an associated access control list (ACL) that lists users and their rights to the object. EDACL edits or displays the ACL of the object(s) specified. The structure and usage of an ACL is described in detail in the *DOMAIN System User's Guide*.

## ARGUMENTS

<b>pathname</b> (required)	Specify the object whose ACL you wish to edit or display. Multiple pathnames and wildcarding are permitted.
<b>commands</b> (optional)	Specify the action(s) described below. If you do not specify a command, EDACL enters an interactive editing mode.  Default if omitted: read commands from standard input; do not precede commands with a hyphen (-) in this mode.

## COMMANDS

Many of the commands described below take arguments called 'sid' and 'rights'. These are summarized in the sections preceeding the EXAMPLES.

<b>-L</b>	List ACL entries.
<b>-A sid rights</b>	Add the specified entry to an ACL. You will receive an error message if the ACL entry exists.
<b>-AF sid rights</b>	Add force. Add the specified entry to an ACL. You will not receive an error message if the ACL entry exists.
<b>-AR sid rights</b>	Add the specified rights to an ACL. You will receive an error message if the entry does not exist.
<b>-C sid rights</b>	Change the access rights in the entry for 'sid' (replaces current rights). You will receive an error message if the entry does not exist.
<b>-CF sid rights</b>	Change force. Change the access rights in the entry for 'sid' (replaces current rights). You will not receive an error message if the entry does not exist.
<b>-D sid</b>	Delete the ACL entry for 'sid'. You will receive an error message if the entry does not exist. If 'sid' is '%.%.%.%', then

EDACL will leave the entry with 'S' and 'E' rights to maintain DOMAIN/IX compatibility.

- DF sid rights**      Delete force. Delete the specified rights from the entry for 'sid'. You will not receive an error message if the ACL entry does not exist.
- DR sid rights**      Delete the specified rights from the entry for 'sid'. You will receive an error message if the entry does not exist.
- CDN node**            Change the default node ID.
- CN sid node**        Change the node ID entry in 'sid'.
- Q**                    Quit without changing the object's ACL. This command is useful only when you supply EDACL commands interactively (see -I). To signal successful completion and update the ACL, use EOF in standard input (usually <CTRL/Z>).

The following three commands are meaningful primarily for DOMAIN/IX applications. If the pertinent index is enabled, the process executing the file assumes the PERSON, PROJECT, and/or ORGANIZATION identity of the file. (This is the DOMAIN/IX equivalent of AEGIS protected subsystems.) The indexes may be set for both files and directories, but are meaningful only for files.

- SETPERS {sid|0}**      Assign the SET PERSON index to 'sid'.  
If you specify '0' (zero) instead of a sid, the SET PERSON index is deleted.
- SETPROJ {sid|0}**     Assign the SET PROJECT index to 'sid'.  
If you specify '0' (zero) instead of a sid, the SET PROJECT index is deleted.
- SETORG {sid|0}**      Assign the SET ORGANIZATION index to 'sid'.  
If you specify '0' (zero) instead of a sid, the SET ORGANIZATION index is deleted.

#### OPTIONS

- DIR**                    Only operate on directories.
- FILE**                 Only operate on files.
- ID**                    Edit the default initial ACL for directories (-DIR implied).
- IF**                    Edit the default initial ACL for files (-DIR implied).
- UNIX**                 Enable editing of 'S' and 'E' rights for directories. This is

meaningful primarily for DOMAIN/IX applications. Modification of these rights is disabled by default, unless this option is specified.

- DYN[AMIC]** Create a dynamic ACL for use with DOMAIN/IX applications. Dynamic ACLs are computed and assigned "on the fly" by DOMAIN/IX programs; thus, they change from user to user rather than remaining static, like AEGIS ACLs. Use of this option precludes the use of any of the editing functions listed above in the "COMMANDS" section.

The following two options apply only when EDACL reads commands from standard input:

- P** EDACL interprets commands when it receives an EOF (usually <CTRL/Z>). This is the default when you have redirected standard input (i.e., instructed the program to read commands from a Shell program, here document, file, or pipe).
- I** EDACL interprets commands as you enter them. This is the default when you have not redirected standard input. You may only specify one pathname (with no wildcards) in this mode. EDACL changes a copy of the ACL; the command does not assign a new ACL to an object until it reads an EOF. Thus, EDACL -I does not change an ACL if you terminate the session with the "Q" command.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### SIDS

An SID (subject identifier) is the mechanism used to identify people to the system when they log in. Basically, an SID has three parts: a person name (P), project name (P), and organization name (O); the combination is often abbreviated to 'PPO'. In some cases, the node on which the subject is running is of importance as well. Thus, a full SID also contains this item of information, in which case it is a 'PPON'; but most of the time PPO is all that is of concern.

SIDs consist of the P,P, and O separated by periods. Thus

joe.sftwr.r\_d

might be the name of a software programmer in the R & D organization. His person name is 'joe'; his project name is 'sftwr'; his organization name is 'r\_d'.

If the node ID is required then a PPON for the above example might look like:

joe.sftwr.r\_d.14C

where '14C' is the node ID of the node where 'joe' is logged in.

In ACLs, SIDs may contain one or more wildcards, similar in concept to wildcards used with pathnames. A '%' in the person, project, organization, or node id part of a SID will match any person, project, organization, or node (respectively). Thus

joe.%.%.%

would match user 'joe' regardless of his project or organization names, and regardless of which node he happened to be using.

### RIGHTS

The following are the basic kinds of operations that can be performed on objects, and the rights which allow them when present in an ACL entry.

for all objects:

```
p    protect rights; allows rights to be changed
g    grant rights; allows creation of new entries
      with a subset of creator's rights
n    change node list rights; allows CD, CN commands
```

for files:

```
d    delete rights; allows file to be deleted
w    write rights; allows file to be written
r    read rights; allows file to be read
x    execute rights
```

for directories:

```
d    delete rights; allows directory to be deleted
c    change rights; allows names to be changed,
      and links to be deleted
a    append rights; allows names to be added to directory
l    link rights; allows links to be added to directory
r    read rights; allows directory to be listed
s    search rights; allows directory to be searched for
      subordinate objects (for DOMAIN/IX)
e    expunge rights; allows subordinate objects to be
      deleted provided delete rights are also available
      for the subordinate object (for DOMAIN/IX)
```

The following abbreviations exist for sets of rights: (Note that search and expunge rights are always set.)

```
-OWNER    Gives all rights.
           For files, it means:  pgndwrx
           For directories:     pgndcalrse
-USER     Gives all rights except ability to change ACL.
           For files, it means:  dwrx
           For directories:     dcalrse
-READ     For files, allows reading; can't change ACL.
           Precisely, it means:  r
-EXEC     For files, allows reading, execution; can't change ACL.
           Precisely, it means:  rx
-LDIR     For directories, allows listing; can't change ACL.
           Precisely, it means:  rse
-ADIR     For directories, allows adding names and links,
           and listing; can't change ACL.
           Precisely, it means:  alrse
-NONE     Gives no rights, for files or directories.
           Used to explicitly deny rights to specific
           SIDs that would otherwise be granted rights
           because they are members of a project or
           organization.
           For directories it means: se (unless -UNIX was
```

## EDACL (EDIT\_ACCESS\_CONTROL\_LIST)

specified when EDACL was invoked, in which case all rights are revoked.)

### EXAMPLES

(The *DOMAIN System User's Guide* also provides detailed examples of applying and manipulating ACLs.)

1. The order of the commands in the following sequence is significant.

\$ edacl -L sales %.%.%.% pgndwrx \$	List ACL for the file 'sales'. The ppon is all wildcards (%.%.%.%), so all users have complete rights (pgndwrx) to 'sales'.
\$ edacl sales -cf dan.% -none \$ edacl -L sales DAN.%.%.% ----- %.%.%.% pgndwrx \$	Deny user DAN access to 'sales'. Other users still have all rights. Note that the system automatically places specific entries before general ones.
\$ edacl sales -a joe -owner \$ edacl -L sales joe.%.%.% pgndwrx dan.%.%.% ----- %.%.%.% pgndwrx \$	Add user JOE to the ACL for 'sales' with all rights.
\$ edacl sales -a %.%.mktg wrx \$ edacl -L sales joe.%.%.% pgndwrx dan.%.%.% ----- %.%.mktg.% ----wrx %.%.%.% pgndwrx \$	Allow users in the MKTG organization to change file contents, but do not let them assign rights to others (p and g), change the node ID entry (n), or delete the file (d).
\$ edacl sales -c % r \$ edacl -L sales joe.%.%.% pgndwrx dan.%.%.% ----- %.%.mktg.% ----wrx %.%.%.% ----r-- \$	Change everyone else's access to read only. Note that the more liberal rights (wrx) assigned to the MKTG organization in the previous line still apply, since specific entries override general ones.

2. The following examples illustrate the effect of the -UNIX option.

```
$ edacl dir
dir
* l
%.%.%.%                pgndcalrse
* a jim -none
jim.%.%.%              -----se
* a ers -r
ers.%.%.%              -----rse
* l
jim.%.%.%              -----se
ers.%.%.%              -----rse
%.%.%.%                pgndcalrse
```

Now specify -UNIX ...

```
$ edacl dir -unix
dir
* l
%.%.%.%                pgndcalrse
* a jim -none
jim.%.%.%              -----
* a ers -r
ers.%.%.%              -----r--
* l
rees.%.%.%             -----
ers.%.%.%              -----r--
%.%.%.%                pgndcalrse
```

3. Set the initial file ACL for the directory //test/tmp/dir to be dynamic.

```
$ edacl //test/tmpdir -if -dyn
```

## EDFONT

**EDFONT -- Edit a character font.**

### FORMAT

**EDFONT [pathname [char]]**

EDFONT is an interactive, menu-driven program that allows you to create, edit and view character font files. For a detailed explanation on editing a character font, see the EDFONT description in the *DOMAIN System Utilities* manual.

### ARGUMENTS

**pathname**  
**(optional)**

Specify the name of the font file to be edited.

Default if omitted: EDFONT prompts for the pathname.

**char**  
**(optional)**

Specify the first character to be edited. This argument is valid only if the 'pathname' argument has been specified.

Default if omitted: begin editing session with character 'g'.

**EDIP (EDIT\_IP\_HELPER) -- Invoke editor for IP\_HELPER.**

## FORMAT

**EDIP [[net.]node\_id]**

EDIP allows you to inspect and modify the IP\_HELPER's directory of DOMAIN node names, Internet Protocol addresses, DOMAIN addresses, and the IP\_HELPER's replica list. Once invoked, EDIP enters an interactive mode and accepts the commands described below.

## ARGUMENTS

**[net.]node\_id**

**(optional)**

Set the default IP\_HELPER to the IP\_HELPER at the node specified by this identifier.

## COMMAND SUMMARY

Some EDIP commands use node specifications and DOMAIN internet addresses as arguments. If a command accepts a node specification, the syntax line uses the term 'node\_spec'. If a command accepts a DOMAIN internet address, the syntax line uses the term '[net.]node\_id'.

When a command accepts a node specification, you can provide a node name (if the name is cataloged in the default IP\_HELPER database) or you can provide a DOMAIN internet address. When a command accepts a DOMAIN internet address, you must specify a node ID, and in some cases, a DOMAIN network number.

Note that the rules for specifying DOMAIN internet addresses in EDIP commands differ slightly from the rules for specifying DOMAIN internet addresses in Shell commands. If the node is on the local ring, the network number is optional. If the node is on a remote ring, the network number is required. When you issue EDIP commands from a node in a network that has a non-zero network number, you cannot specify the number 0 to indicate the local ring.

When you specify a DOMAIN internet address in an EDIP command, the DOMAIN internet address must begin with an integer. If the address begins with a letter, precede the address with a 0 (e.g. 0D34.1E05).

## EDIP (EDIT\_IP\_HELPER)

---

SYNTAX (abbreviation shown in UPPER case)	FUNCTION
<hr/>	
Default options are indicated by "(D)".	
ADD name -IP Internet_Protocol_address -DA [net.]node_id	Adds a DOMAIN node name, its Internet Protocol(IP) address, and its DOMAIN address to the default IP_HELPER's copy of the directory. The IP address must be entered as 4 decimal digits each separated by a period ".". The DOMAIN address net and nodeid must be entered as hexadecimal digits separated by a period ".". The DOMAIN address or the IP address may be omitted.  IP_HELPER propagates the new information to all IP_HELPERS in its replica list. If the node name or its IP address already exists in IP_HELPER's database, EDIP does not add the entry and displays an error message.
ADDRep node_spec	Adds a replica to the default IP_HELPER's replica list. The IP_HELPER propagates the new replica's identity to all IP_HELPERS in its replica list.  The IP_HELPER accepts a new replica only if the entry does not already exist in the replica list. If an entry already exists, then the IP_HELPER does not accept the entry and EDIP displays an error message.
CHRep Old_node_spec New_node_spec	Changes the DOMAIN address of the replica old_node_spec to new_node_spec in the default IP_HELPER's replica list; only the DOMAIN net number may be changed. The IP_HELPER propagates the change to all IP_HELPERS in its replica list, including the replica whose address changed.
CMP [node_specA] node_specB [-LA -ND]	Compares two IP_HELPER directories and lists entries that appear in both directories. The CMP command shows names and Internet Protocol addresses that appear in both copies of the IP_HELPER directory. If you do not provide a value for node_specA, then EDIP uses the default IP_HELPER database.

- ND** (D) Do not list entries that are exact duplicates. Only list cases where a name or Internet Protocol address is associated with different entry information in the two directories.
- LA** List all names and Internet Protocol addresses that appear in both directories.
- DELEte name** Deletes a node name from the default IP\_HELPER's copy of the directory. The IP\_HELPER propagates the delete request to all IP\_HELPERS in its replica list. If the name you specify does not exist, EDIP returns an error message and does not accept or propagate the DELETE request.
- DELRep node\_spec** Deletes an IP\_HELPER from the default IP\_HELPER's replica list. The IP\_HELPER propagates the delete request to all IP\_HELPERS in its replica list, thereby removing the replica from all other replica lists. In addition, DELRep causes the deleted replica to delete its database. The deleted replica stops running after its propagation list has been emptied.
- If the replica you specify with the DELRep command does not exist in the IP\_HELPER's replica list, EDIP returns an error and does not accept or propagate the DELRep request.
- It is best to wait at least fifteen minutes before restarting a deleted replica.
- DIFF [node\_specA] node\_specB** Lists the differences between two IP\_HELPER databases. The DIFF command shows differences between the copies of the IP\_HELPER directory and between the replica lists. If you do not provide a value for node\_specA, then EDIP uses the default IP\_HELPER database.
- INFO** Displays the DOMAIN address and status information for the default IP\_HELPER.

## EDIP (EDIT\_IP\_HELPER)

INIT [node\_specA] [-FROM node\_specB]

Initializes an IP\_HELPER database. If you do not specify a value for node\_specA, then EDIP initializes the default IP\_HELPER database. After you initialize an IP\_HELPER it becomes active. That is, the IP\_HELPER can communicate with other IP\_HELPERS and can respond to lookup and update requests. (Before an IP\_HELPER is initialized, it will respond only to the INFO, INIT, MERGE\_ALL and SHUT commands.)

-FROM node\_specB

If you specify the -FROM option, EDIP performs some additional initialization.

First, EDIP adds the IP\_HELPER on node A (or the default node) to node B's replica list. Then node B propagates the new replica information to all the replicas in its (node B's) replica list. Thus, the other IP\_HELPERS will now have node A (or the default IP\_HELPER) on their replica lists.

Finally, EDIP merges all entries from node B's IP\_HELPER database into node A's (or the default) database. This merge includes the entries in node B's copy of the directory and in node B's replica list.

LD [names] [-NODE node\_id]  
[-NET net]  
[-IPADDR IP\_address]  
[-SN|-NSN]  
[-N] [-DA] [-IP] [-DTE]

Lists IP\_HELPER directory entries by name; if names are specified, only those names are listed.

-NET net

Lists directory entries with the specified DOMAIN network number. If names are specified, LD lists entries with the specified names, and also lists entries with the specified network number.

-NODE node\_id

Lists entries with specified DOMAIN node ID. If names are also specified, LD lists entries with the specified names and also lists entries with the specified node ID.

-IPADDR IP\_address

Lists entries with specified Internet Protocol address. If names are also specified, LD lists entries with the specified names and also lists entries with the specified IP address.

-SN  
-NSN

(D)

Lists entries sorted by name.  
Suppresses name sorting.

The following options specify what special information should be displayed with each entry that is listed:

-N Displays node\_id.  
 -DA Displays DOMAIN address.  
 -IP Displays Internet Protocol(IP) address.  
 -DTE Displays date/time this entry was made to the directory and the node\_id of the replica where this entry was made.

LR [-CLOCKS] Displays list of replicas in the network.

-CLOCKS Displays each replica's current clock date/time and checks for any replicas whose clocks are skewed.

MERGE [node\_specA] -FROM node\_specB

Merges all entries in the IP\_HELPER database on node B into the IP\_HELPER database on node A; node B's database remains unchanged. If you do not specify a value for node A, then EDIP merges node B's database into the default IP\_HELPER database.

If node A's database contains an entry with the same name as an entry being merged from node B, then the entry with the latest timestamp is saved in node A's database. (A timestamp is the time an entry receives when it is first added to an IP\_HELPER database. An entry keeps its original timestamp when it is propagated to other IP\_HELPERS.)

The MERGE command affects only the database on node A; node A does not propagate any entries it obtains from the merge.

MERGE\_ALL [node\_spec]

Performs a global merge using the IP\_HELPER at the node you specify as the base for the merge. If you omit the node\_spec, EDIP uses the default IP\_HELPER.

To do a global merge, EDIP merges each IP\_HELPER database in the specified IP\_HELPER's replica list into the specified IP\_HELPER's database. Then, EDIP merges the updated database back out to each replica. EDIP merges both the replica lists and the copies of the directory. If a database contains an entry with the same name as an entry being merged, then the entry with the latest timestamp is saved. (A timestamp is the time an entry receives when it is first

## EDIP (EDIT\_IP\_HELPER)

added to an IP\_HELPER database. An entry keeps its original timestamp when it is propagated to other IP\_HELPERS.)

An IP\_HELPER must be listed in the base IP\_HELPER's replica list to be included in a global merge. The IP\_HELPERS in the replica list may be uninitialized. If an IP\_HELPER is not already initialized, the MERGE\_ALL command will initialize its database and allow the IP\_HELPER to be active.

QUIT

Terminates the current EDIP session.

REPlace name -IP IP\_address  
-DA [net.]node\_id

Changes the Internet Protocol(IP) address and DOMAIN address associated with name in IP\_HELPER's copy of the directory. The IP address must be entered as 4 decimal digits each separated by a period (.). The DOMAIN address net and nodeid must be entered as hexadecimal digits separated by a period. The DOMAIN address or the IP address may be omitted; an address which is omitted is entered as NULL for the entry.

IP\_HELPER propagates the new information to all IP\_HELPERS in its replica list. If the IP address is associated with another entry that already exists in IP\_HELPER's database, EDIP does not replace the entry and displays an error message.

SET [node\_spec]

Sets the default IP\_HELPER to the IP\_HELPER running on the node you specify. Subsequent EDIP commands will be directed to this IP\_HELPER, unless you specify a different IP\_HELPER in the command. If you use the SET command and omit a node specification, EDIP will select an active IP\_HELPER (with an initialized database) to be the default.

SHUT node\_spec

Shuts down the IP\_HELPER replica on the node you specify. This command causes the specified replica to delete its database and stop running immediately. The shutdown replica is not removed from any replica lists.

For complete information on EDIP command usage, see *Managing TCP/IP-Based Communications Products*.

**EDMTDESC (EDIT\_MAGTAPE\_DESCRIPTOR) -- Edit magtape descriptor file.**

## FORMAT

**EDMTDESC pathname {options}**

EDMTDESC allows you to create, list, and modify magnetic tape descriptor objects. These descriptor files provide information to the streams manager so that it can handle subsequent tape operations much as an SIO descriptor file describes the configuration of an SIO line.

## ARGUMENTS

<b>pathname</b> (required)	Specify name of magtape descriptor file to be created, listed, or edited.
-------------------------------	---

## OPTIONS

At least one of the following options must be specified.

<b>-C</b>	Create a new magtape descriptor object with the name given in the 'pathname' argument.
<b>-L [var...]</b>	List the values of the variable(s) specified. If no variables are named, the entire magtape descriptor is listed.
<b>-S {var value}...</b>	Set the variable(s) indicated to the specified value(s). At least one variable/value pair is required if -S is specified. Multiple variable/value pairs are permitted, separated by blanks.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## VARIABLES

The variables known to EDTDESC are listed below, along with their types and default values. The variable types are: integer (int), Boolean (y/n), character string of n letters (c [n]), and date (in format yy/mm/dd.hh:mm).

## EDMTDESC (EDIT\_MAGTAPE\_DESCRIPTOR)

name	type	default	definition
DEV	c[1]	M	device type ('M' for magtape, 'C' for cartridge)
U	int	0	magtape unit number (normally 0)
LAB	y/n	yes	'yes' if magtape is ANSI labeled, 'no' if unlabeled
REO	y/n	no	'yes' to reopen previously used volume, 'no' to open new volume ('yes' suppresses rewind)
CLV	y/n	yes	'yes' closes volume when file is closed, 'no' leaves volume open
SPOS	y/n	no	'yes' saves volume position when volume is closed (for reopen), 'no' rewinds volume when closed
VID	c[6]	-auto	volume identifier (labeled volumes)
VACC	c[1]		volume accessibility (labeled volumes)
OWN	c[14]	-auto	volume owner (labeled volumes)
F	int*	1	file sequence number -- integer or "cur" for current file, or "end" for new file at end of labeled volume
RF	c[1]	D	record format -- "F" for fixed length, "D" for variable length, "S" for spanned, "U" for undefined
BL	int	2048	block length, in bytes
RL	int	2048	(maximum) record length, in bytes
ASCNL	y/n	yes	'yes' for ascii newline handling (strip newlines on write, supply them on read), 'no' for no newline handling
FSECT	int	1	file section number (labeled volumes)
FID	c[17]		file identifier (labeled volumes)
FSID	c[6]		file set identifier (labeled volumes)
GEN	int	1	generation of file (labeled volumes)
GENV	int	1	generation version of file (labeled volumes)
CDATE	date	-auto	creation date of file (labeled volumes)
EDATE	date	-auto	expiration date of file (labeled volumes)
FACC	c[1]		file accessibility (labeled volumes)
SYSC	c[xx]		system code (labeled volumes)
SYSU	c[xx]		system use (labeled volumes)
BOFF	int	0	buffer offset (labeled volumes, should be 0)

For cartridge tape (DEV C), you must change the block length (BL) and the record length (RL) to be 512 or less and the record format to be fixed ('RF F').

### EXAMPLES

```
$ edmtdesc set_tape -s u 1 lab yes
```

```
Edit file "set_tape";
set the tape unit number
to 1; declare tape as
ANSI labeled.
```

```
$ edmtdesc ct -c -s dev c bl 512 rl 128 rf f
```

```
Create descriptor file ct for
cartridge tape, blocking 4
records of maximum length 128
to each block.
```

**EDNS (EDIT\_NAMING\_SERVER\_HELPER) -- Invoke editor for NS\_HELPER.**

## FORMAT

**EDNS [[net.]node\_id]**

EDNS allows you to inspect and/or modify NS\_HELPER's master network root directory and replica list. Once invoked, EDNS enters an interactive mode and accepts the commands described below. For complete information on NS\_HELPER and EDNS usage, see *Administering Your DOMAIN System*.

## ARGUMENTS

**[net.]node\_id**  
(optional)

Set the default NS\_HELPER to the NS\_HELPER at the node specified by the internet address.

Default if omitted: Set the default NS\_HELPER to any active NS\_HELPER. An NS\_HELPER becomes active after its database has been initialized.

## EDNS COMMAND SUMMARY

Some EDNS commands use node specifications and internet addresses as arguments. If a command accepts a **node specification**, the syntax line uses the term *node\_spec*. If a command accepts an **internet address**, the syntax line uses the term *[net.]node\_id*.

When a command accepts a node specification, you can provide a node name (if the name is cataloged in the default NS\_HELPER database) or you can provide an internet address.

When a command accepts an internet address, you must specify a node ID, and in some cases, a network number. Note that the rules for specifying internet addresses with EDNS commands differ slightly from the rules for specifying internet addresses with Shell commands. If the node is on the local ring, the network number is optional. If the node is on a remote ring, the network number is required. When you specify an internet address with an EDNS command, the internet address must begin with an integer. If the address begins with a letter, precede the address with a 0 (e.g., 0D34.1E05). When you issue EDNS commands from a node in a non-zero network, you cannot specify the number 0 to indicate the local ring.

For complete information on EDNS command usage, see *Administering Your DOMAIN System* and *Managing DOMAIN Internets*.

SYNTAX (abbreviation shown in uppercase)	FUNCTION
Default options are indicated by "(D)".	
ADD name [net.]node_id	<p>Adds a new node name, and the associated node information, to the default NS_HELPER's copy of the master root directory. After accepting a name, the NS_HELPER propagates the new information to all NS_HELPERS in its replica list. The NS_HELPER accepts a new name only if it does not already exist in the master root directory. If a name already exists, then the NS_HELPER does not accept the entry and EDNS displays an error message.</p> <p>The node you are adding must be connected to the network in order for EDNS to obtain information needed for the database. For a disked node, EDNS gets the UID for the node entry directory. For a diskless node, EDNS verifies that it is diskless and creates a canned UID.</p>
ADDRep node_spec	<p>Adds a replica to the default NS_HELPER's replica list. The NS_HELPER propagates the new replica's identity to all NS_HELPERS in its replica list.</p> <p>The NS_HELPER accepts a new replica only if the entry does not already exist in the replica list. If an entry already exists, then the NS_HELPER does not accept the entry and EDNS displays an error message.</p>
CMP [node_specA] node_specB	<p>Compares two NS_HELPER databases and lists names, network numbers, and UIDs that appear in both copies of the master root directory. CMP also lists replicas that appear in both replica lists. If you do not provide a value for node_specA, then EDNS uses the default NS_HELPER database.</p>
DELEte name	<p>Deletes a node name from the default NS_HELPER's copy of the master root directory. The NS_HELPER propagates the delete request to all NS_HELPERS in its replica list. If the name you specify does not exist, EDNS returns an error message and does not accept or propagate the DELETE request.</p>

## EDNS (EDIT\_NAMING\_SERVER\_HELPER)

DELRep node\_spec

Deletes an NS\_HELPER from the default NS\_HELPER's replica list. The NS\_HELPER propagates the delete request to all NS\_HELPERS in its replica list, thereby removing the replica from all other replica lists. In addition, DELRep causes the deleted replica to delete its database. The deleted replica stops running after its propagation list has been emptied.

If the replica you specify with the DELRep command does not exist in the NS\_HELPER's replica list, EDNS returns an error and does not accept or propagate the DELRep request.

It is best to wait at least fifteen minutes before restarting a deleted replica.

DIFF [node\_specA] node\_specB

Lists the differences between two NS\_HELPER databases. The DIFF command shows differences between both copies of the master root directory, and between both replica lists. If you do not provide a value for node\_specA, then EDNS uses the default NS\_HELPER database.

INFO

Displays the internet address and status information for the default NS\_HELPER.

INIT [node\_specA] [-FROM node\_specB]

Initializes an NS\_HELPER database. If you do not specify a value for node\_specA, then EDNS initializes the default NS\_HELPER database. After you initialize an NS\_HELPER, it becomes active. That is, the NS\_HELPER can communicate with other NS\_HELPERS and can respond to naming requests from other nodes. (Before an NS\_HELPER is initialized, it will respond only to the INFO, INIT, MERGE\_ALL and SHUT commands.)

To use the INIT command without the -FROM option, you must use EDNS from a node on the same ring as the NS\_HELPER you are initializing. In such a case, EDNS gets a list of all nodes on the local ring, and adds these nodes to the NS\_HELPER database that you are initializing. The NS\_HELPER propagates the new information to all replicas in its replica list.

## EDNS (EDIT\_NAMING\_SERVER\_HELPER)

If you initialize an NS\_HELPER that has previously been initialized, the INIT command adds any new node names to the existing database and propagates these names to the NS\_HELPERS on its replica list.

**-FROM node\_specB**

If you specify the **-FROM** option, EDNS performs a different type of initialization. First, EDNS adds the NS\_HELPER on node A (or the default node) to node B's replica list. Then node B propagates the new replica information to all the replicas in its (node B's) replica list. Thus, the other NS\_HELPERS will now have node A (or the default NS\_HELPER) on their replica lists.

Next, EDNS merges all entries from node B's NS\_HELPER database into node A's (or the default) database. This merge includes the entries in node B's copy of the master root directory and in node B's replica list.

Finally, if node A (or the default node) and node B are on different rings, EDNS also gets a list of all nodes on node A's network and adds these nodes to node A's copy of the master root directory. Then node A's NS\_HELPER propagates these names to all the NS\_HELPERS on its replica list. When node A and node B are on different rings, you must use EDNS from a node on the same ring as node A.

**LD [names] [-NODE node\_id]  
[-NET net]  
[-SN|-NSN]  
[-T] [-U] [-N] [-IA] [-DTE]**

Lists root directory entries by name; if names are specified, only those names are listed.

**-NET net**

Lists root directory entries with the specified network number; if names are specified, LD lists entries with the specified names, and also lists entries with the specified network number.

**-NODE node\_id**

Lists entries with specified node ID; if names are also specified, LD lists entries with the specified names and also lists entries with the specified node ID.

**-SN (D)  
-NSN**

Lists entries sorted by name.  
Suppresses name sorting.

## EDNS (EDIT\_NAMING\_SERVER\_HELPER)

The following options specify the special information to be displayed with each entry that is listed:

- T Displays entry type.
  - U Displays UID.
  - N Displays node\_id.
  - IA Displays internet address.
  - DTE Displays date/time this entry was made to the directory and the node\_id of the replica where this entry was made.
- LR [-CLOCKS] Displays list of replicas in the network.
- CLOCKS Displays each replica's current clock date/time and checks for any replicas whose clocks are skewed.
- MERGE [node\_specA] -FROM node\_specB
- Merges all entries in the NS\_HELPER database on node B into the NS\_HELPER database on node A; node B's database remains unchanged. If you do not specify a value for node A, then EDNS merges node B's database into the default NS\_HELPER database.
- If node A's database contains an entry with the same name as an entry being merged from node B, then the entry with the latest time stamp is saved in node A's database. (A time stamp is the time an entry receives when it is first added to an NS\_HELPER database. An entry keeps its original time stamp when it is propagated to other NS\_HELPERS.)
- The MERGE command affects only the database on node A; node A does not propagate any entries it obtains from the merge.
- MERGE\_ALL [node\_spec]
- Performs a global merge using the NS\_HELPER at the node you specify as the base for the merge. If you omit the node\_spec, EDNS uses the default NS\_HELPER.
- To do a global merge, EDNS merges each NS\_HELPER database (in the specified NS\_HELPER's replica list) into the specified NS\_HELPER's database. Then, EDNS merges the updated database back out to each replica. Note that EDNS merges both the replica lists and the copies of the master root directory. If a database contains an entry with the

## EDNS (EDIT\_NAMING\_SERVER\_HELPER)

same name as an entry being merged, then the entry with the latest time stamp is saved. (A time stamp is the time an entry receives when it is first added to the NS\_HELPER database. An entry keeps its original time stamp when it is propagated to other NS\_HELPERS.)

An NS\_HELPER must be listed in the base NS\_HELPER's replica list to be included in a global merge. The NS\_HELPERS in the replica list may be uninitialized. If an NS\_HELPER is not already initialized, the MERGE\_ALL command will initialize its database and allow the NS\_HELPER to be active.

QUIT

Terminates the current EDNS session.

REPLace name [net.]node\_id

Changes the internet address and UID associated with a name in the default NS\_HELPER's copy of the master root directory. The NS\_HELPER propagates the new information to all NS\_HELPERS in its replica list. Use this command after running the utilities CHUVOL, or INVOL, or replacing a disk.

The node you are replacing must be connected to the network in order for the NS\_HELPER to obtain information needed for the database. For a disked node, EDNS obtains the UID for the node entry directory. For a diskless node, EDNS verifies the diskless status and creates a canned UID.

SET [node\_spec]

Sets the default to the NS\_HELPER running on the node you specify. Subsequent EDNS commands will be directed to this NS\_HELPER, unless you specify a different NS\_HELPER in the command. If you use the SET command and omit a node specification, EDNS will select an active NS\_HELPER (with an initialized database) to be the default.

SHUT node\_spec

Shuts down the NS\_HELPER replica on the node you specify. This command causes the specified replica to delete its database and stop running immediately. The shutdown replica is not removed from any replica lists.

## EDNS (EDIT\_NAMING\_SERVER\_HELPER)

UPDATE [node\_spec]

Updates the NS\_HELPER database on the node specified. If you do not specify a node, then EDNS updates the default NS\_HELPER database. An NS\_HELPER must be initialized before it can be updated.

EDNS gets a list of all nodes on the EDNS node's local ring. It adds nodes which are not already in the NS\_HELPER database and replaces node information (such as internet address and entry directory UID) which has changed. NS\_HELPER propagates new information to all replicas in its replica list.

EDPPO (EDIT\_PPO\_FILE)

EDPPO (EDIT\_PPO\_FILE) -- Edit/list person, project, or organization names.

## FORMAT

### EDPPO {options}

EDPPO is used to define usernames in the network registry. This is a prelude to creating network accounts, which associate usernames with log-in home directories and passwords. Use EDACCT (EDIT\_ACCOUNT) to perform that operation.

While all of the available EDPPO options are described below, you will probably not be able to edit network registry files unless you are the network administrator for your network. You should, however, be able to list the contents of the network and local registries. Use EDPPO to find out who is associated with a particular username and account.

For detailed examples of editing network registry files and complete information on network registry commands, see *Administering Your DOMAIN System*.

## OPTIONS

If you omit the -A, -D, -L, and -LF options, EDPPO enters an interactive editing session which reads commands from standard input as described in the COMMANDS section below.

Default options are indicated by "(D)."

**-R pathname** Specify the registry to be edited or listed. You should only use this option with -LOC (described below) to edit or list a remote node's local registry. If you want to edit or list the master registry, omit this option and let EDPPO use the network registry file copy ('/registry/registry') on the current node to locate the master registry.

Only one of the following three options may be present on a command line.

**-PERS (D)** Edit or list the contents of the Person file.

**-PROJ** Edit or list the contents of the Project file.

**-ORG** Edit or list the contents of the Organization file.

**-LOC** Specify that the node's local registry ('/registry/local\_registry') is to be listed. If you specify -LOC, the only other valid options are -R -L, -COL, and -D. You may not manipulate the local registry using EDPPO's interactive mode; the list or delete functions must be specified on the command line.

**-A name [fullname]**

Add a name and optional full name text. PPO names can be up to 32 characters long, must begin with a letter, and can include only letters, numbers, and underscores (\_). PPO names are automatically maintained in lowercase. Associated full name text is optional but strongly recommended. It can include any

characters and be up to 32 characters long. Use single (or double) quotation marks to embed spaces (or quotation marks) in a full name.

- D name** Delete a name. Command line delete is valid only on a local registry. To delete a name from the network registry file, use EDPPO's interactive mode. **DELETE WITH EXTREME CAUTION**, even locally, because once you delete a name no one can **EVER** access any files created by the user who had that PPO. (The system cannot recreate a unique name identifier once it has been deleted.) Normally, you should delete people's accounts (using EDACCT), **NOT** their names.
- L [name...]** List the name(s) specified. If 'name' is omitted, all names are listed.
- LF [name ...]** List the name(s) specified, along with associated full name text, if any. If 'name' is omitted, all names are listed.
- COL** List the names in a single column.
- NP** Suppress prompts during interactive editing.

## COMMANDS

If you omit the -A, -D, -L, and -LF options on the command line, EDPPO enters an interactive editing session which accepts the following commands from standard input.

- a [name [fullname]]**  
Define one or more new names, along with optional full name. If 'fullname' is omitted, EDPPO prompts you for it. Enter an empty line if you wish the fullname field to be null. (See -A above for name formats and restrictions.) If no name is specified, names are read from standard input. We strongly recommend that you supply full names.
- c name new\_name [new\_fullname]**  
Change 'name' to 'new\_name'. If 'new\_fullname' is also specified, change the full name associated with 'name' to 'new\_fullname'.
- cf name new\_fullname**  
Change the full name associated with the name indicated.
- d name**  
Delete the name indicated. Once deleted, the unique name identifier cannot be recreated, so **DELETE WITH EXTREME CAUTION**. Once you delete a name, no one can **EVER** access any files created by the user who had that PPO. Normally, you should delete people's accounts (using EDACCT), **NOT** their names.
- l [name]** List one or all names defined (including new ones).
- lf [name]** List one or all names defined with their associated full names, if any.

## EDPPO (EDIT\_PPO\_FILE)

<b>ln</b>	List only those names added during this editing session.
<b>wr</b>	Update the file with the changes and exit.
<b>q</b>	Quit without updating.
<b>h [comm]</b>	Help -- list briefly all the available commands or a particular command in detail.

### EXAMPLES

List the names of people in the local registry.

```
$ edppo -loc -l
adm      alan      apgar     beth      bls
bso      burt      cas       charlie   chris
chrissy  cmt       col       color     csa
sqh      sys_person taylor    todd      user
vic      wilson_j  zahn
```

**EDSTR (EDIT\_STREAM) -- Edit a stream.**

## FORMAT

**EDSTR [-N] { command | -E command | -F cmdfile ...} [pathname]**

EDSTR copies the named input files to standard output, performing editing as directed by EDSTR commands in the command line or in the named command file.

## ARGUMENTS

If neither the -E or -F argument is specified, EDSTR assumes that the first token on the command line without a hyphen is an EDSTR command (see below) and that the remaining tokens (if any) are pathnames.

**command**  
(optional)

Specify a single EDSTR command (except A, C, or I). EDSTR accepts the ED commands A, C, D, I, P, R, S, W, and =. To use the A, C, or I commands, place them in a command file as described below.

Default if omitted: use -E and/or -F

The following two arguments may be repeated and intermixed in any order. EDSTR executes them in the order in which they appear on the command line.

**-E command**  
(optional)

Specify an EDSTR command (except A, C, or I). To use the A, C, or I commands, place them in a command file as described below. EDSTR can accommodate commands totaling approximately 5000 characters (including <text> arguments), and lines up to 120 characters long.

Default if omitted: use 'command' or -F

**-F cmdfile**  
(optional)

Specify a file containing EDSTR commands, one per line. Control is passed to this file for command processing. See -E for EDSTR command restrictions.

Default if omitted: use 'command' or -E

**pathname**  
(optional)

Specify input file to be edited. Multiple pathnames are permitted.

Default if omitted: edit standard input

## OPTIONS

**-N**

Suppress writing of output except for P and W EDSTR commands. By default, EDSTR writes each line of input to standard output after editing. If this option is specified, it must precede any arguments on the command line.

## EDSTR (EDIT\_STREAM)

### EXAMPLES

```
$ edstr -e s/joe/mary/g -e '20r add_stuff' infile > outfile
```

This command first replaces all occurrences of "joe" with "mary", then copies material in the file "add\_stuff" into "infile" following line 20. Results are written to the file "outfile".

### SUMMARY OF EDSTR COMMANDS

#### Addresses:

```
17      a decimal number
$       the last line of the file
/pat/   search forward for line containing pat
\pat\   search backward for line containing pat
line+n  n lines forward from line
line-n  n lines backward from line
```

#### Defaults:

```
()      (no address) use current line
(+1)    use the next line
(1.$)   use all lines
```

#### Commands:

```
()      A          Append text after line (text follows)
()      C          Change text (text follows)
()      D          Delete text
()      I          Insert text before line (text follows)
()      P          Print text (can be appended to other commands)
()      R file     Read file, appending after line
()      S/pat/new/GP Substitute new for leftmost pat (G implies all occurrences)
(1.$)   W file     Write file, leave current text unaltered (if no file is specified, write to current filename)
()      =[P]      Print line number, current line
```

#### Arguments:

```
$n      Write to/read from the nth temporary buffer
```

**EM3270 (EMULATE\_3270) -- Emulate an IBM 3270 terminal.**

## FORMAT

**EM3270.{device}**

EM3270 allows a DOMAIN node to emulate an IBM 3270 terminal over an SIO line connected to a VT100-to-3270 converter. The command is meaningless without this additional hardware.

While EM3270 requires no arguments or options, there are actually three different commands, depending on which protocol converter you are using. The following protocol converters support the EM3270 Package software:

- ICCI Model CA20
- ICCI Model CA12
- KMW Model BAC-3270 FS
- PCI 1076

Specify the device name with the EM3270 command. For example:

```
$ em3270.pci
```

if you are using the PCI 1076 protocol converter.

Follow the manufacturer's directions for connecting the converter you choose to the node's SIO lines.

### *EM3270 Commands*

Once you have invoked EM3270, you may use the following commands:

- |          |  |
|----------|--|
| H        | Display command summary information.   |
| LI [n]   | Select SIO line n. The default SIO line is 1.  |
| Q        | Exit from EM3270.  |
| SPEED n  | Set SIO line speed. Valid speeds are 50, 75, 110, 134, 150, 300, 600, 1200, 2000, 2400, 3600, 4800, 7200, 9600, and 19200. |
| [NO]SYNC | Enable/disable XON/XOFF on the SIO line.   |

In addition to these commands, two control/key sequences perform special functions:

- |           |   |
|-----------|---|
| CTRL/<F8> | Switch between command mode and Remote 3270 mode. |
| CTRL/<F7> | Display a layout of the 3270 emulation keyboard.  |

*Keyboard Conversion*

The following special keyboard keys map to the IBM equivalents indicated.

HEX CODE	IBM KEYBOARD	APOLLO KEYBOARD
X'5F'	CENT SIGN	LEFT BRACKET '['
X'4A'	NOT SIGN (PLI-NOT)	RIGHT BRACKET ']'
X'6A'	DOUBLE VERTICAL BAR (ONE ABOVE THE OTHER)	CARET '^'
X'4F'	VERTICAL BAR (PLI-OR)	DOUBLE VERTICAL BAR

**EMT (EMULATE\_TERMINAL) -- Emulate a dumb terminal.**

## FORMAT

**EMT [pathname]**

EMT allows your node to emulate an ASCII terminal connected to another computer. This asynchronous connection exists through a stream opened on one of the node's SIO lines. EMT also permits ASCII file transfer between your node and the remote host.

## ARGUMENTS

**pathname**  
(optional)

Specify file containing EMT commands.

Default if omitted: read commands from standard input

### *Using EMT*

EMT begins execution in local mode, and displays the following prompt:

```
emt>
```

To enter remote mode, press <F1>. (The EMT command DL no longer exists.) In remote mode, your terminal operates as if it is physically connected to the remote computer ("host"). You can log on and enter remote host commands.

To return to local mode, press <F1> again.

### *Input/Output Streams*

EMT uses the four standard streams (standard input, standard output, error input, and error output) as follows:

- EMT commands are read from an EMT command file or from standard input. The command filename may be specified on the command line or using the EMT 'run' command. Up to four levels of command files may be nested. When EOF is reached on a command file, commands are read from the previous file or from standard input. If EOF is reached on standard input, EMT exits.
- Keystrokes to be sent to the host computer are read from error input. Error input may not be redirected to a file. Use the EMT 'xmit' command to transmit a file (of commands or data) to the host. Use the EMT 'rcv' command to receive host transmissions to a DOMAIN file.
- EMT Command responses and all messages from the host are written to standard output.
- Error messages from AEGIS system calls are written to error output. Optional monitoring (MONIT) may also be written to error output (or to a named file).

## EMT (EMULATE \_ TERMINAL)

### *Transferring Files*

You can transfer files using EMT's receive (RCV) or transmit (XMIT) commands. XMIT sends a DOMAIN file to the remote host. RCV opens a DOMAIN file to receive information from the remote host. For example, if you type (in local mode):

```
emt> XMIT FILEA
```

EMT displays the following message:

```
Ready to transmit file FILEA
```

Next, press <F1>. EMT enters remote mode, and transmits FILEA to the remote host.

If you type:

```
emt> RCV FILEB
```

EMT displays this message:

```
Ready to receive file FILEB.
```

Next, enter remote mode by pressing <F1>. Use a remote host command to display the information that you want FILEB to receive. EMT automatically writes this and all subsequent host transmissions into FILEB. To stop the RCV, press <F2>.

### *Transmission Conventions*

Use the EMT command INTERM to specify the line terminator used by the host. If you do not know what the host uses as a line terminator, experiment by changing INTERM. Use the EMT command OUTTERM to specify the line terminator to be transmitted to the host.

EMT allows you to open only one DOMAIN file at a time. If EMT receives a XMIT or RCV command while another DOMAIN file is active, it closes the open DOMAIN file, and executes the new command.

During remote mode, EMT waits on both the keyboard and SIO line for characters to process, and monitors the data for characters of special interest to EMT.

You can specify which keyboard characters EMT should interpret by placing the keyboard in raw or cooked mode. In raw mode, EMT passes all keyboard input (except the function keys, keys L1 through L12, and keys R1 through R4), directly to the host. Cooked mode lets you use many of the Display Manager's features for editing the input pad. EMT places your keyboard in cooked mode by default.

## EMT COMMANDS

*Keys*

- <F1> Switch between local and remote modes.
- <F2> Interrupt a file transfer and close the file.
- <F3> Turn TEE on or off. TEE on causes EMT to display file transmission records on the screen. You can use this feature to monitor file transfers, and decide if and when you should stop or interrupt a transfer. The default is TEE on.
- <F8> Send a BREAK to the host.
- CTRL/<F7> Display function key definitions.

These function keys may be simulated by typing the EMTEESC character followed by the function key number (i.e., ~1 for F1). When EMT is used from the VT100 emulator, F1-shifted is used instead of F2, and F1-control is used instead of F3.

*Commands*

- AE Abort on error.
- ASOnly | NOTASOnly  
Sift out most non-printing ASCII codes. Eliminates triangles, allows BS, CR, ESC, FF, LF, TAB. The default is NOTASC.
- BREAK [n] Set the BREAK duration value to n milliseconds. The default is 200. If set to 0, the <F8> (break) key does nothing.
- CLOSE Deactivate an RCV file. See the RCV command for related information.
- CODE [xx | NONE]  
Set the HOST-COMMAND-CODE to the hexadecimal number xx. The default is NONE.
- COOKED Place the keyboard in cooked mode. This enables many Display Manager features for editing the input pad, and provides an escape sequence for sending control characters to the remote host. To send the host a CTRL character, precede the character with a tilde (~). The sequence ~\_ transmits a delete character. To send the host a single tilde character, type ~~.

The EMT default is cooked mode. Cooked mode always echos keystrokes, so it does not require a full duplex connection to the host. (See the RAW command for related information.) Note that the COOKED and RAW commands refer only to the transcript pad and keyboard input. The SIO line itself is always in RAW mode.

- EMTEESC [chr|NONE]  
Set the EMT escape character to chr. Use NONE to disable the escape character. Default is ~ for cooked mode, NONE for raw mode.

## EMT (EMULATE\_TERMINAL)

The following three commands are useful when standard input is redirected to a file of EMT commands:

- F1** Enter remote mode (Simulate Function key F1).
- F2** Terminate file transfer (Simulate Function key F2).
- F3** Toggle TEE mode (Simulate Function key F3).
- HANGUP** Cause modem to break connection with the remote host.
- HELP [tctl]** Display information about EMT commands or about TCTL commands.
- LINE {1|2|3|pathname}**  
Select the SIO line. Pathname must specify an SIO device descriptor (e.g., /DEV/SIO2). The default SIO line is 1 (/DEV/SIO1).
- L** Display the current SIO line, all EMT switch settings and the receive filename, if any.
- MONIT [pathname]**  
Write every character received over the SIO line to 'pathname'. If a filename is not specified, the previous specification or error output is used.
- NOMONIT** Stop monitoring.
- QUIT** End the EMT session.
- RAW [-ECHO|-NOECHO] [-LF]-NOLF]**  
Place the keyboard in raw mode. This sends keyboard input directly to the remote host, interpreting only function keys. The -ECHO option echos keystrokes on standard output; you should use it when the host is in half-duplex mode. The default is -NOECHO. The -LF option converts CR to LF for lines echoed. The default is -NOLF. (See the COOKED command for related information.) Note: The -ECHO and -LF options are purely local functions that enable you to read what you type. They do not in any way change host/node transmissions.
- RCV [-R] [-KEYS|-NOKEYS] [pathname]**  
Prepare the DOMAIN file specified to receive remote host transmissions. If 'pathname' already exists, EMT appends the transmission to it, unless you specify -R. The receive begins when you enter remote mode <F1>. If you omit 'pathname', EMT uses the previous name, if any. The -KEYS option writes keystrokes to the file along with received data. The default is -NOKEYS.

EMT allows you to interrupt an RCV command at any time by pressing <F2>. EMT remains in whatever mode it was in, but keeps the RCV file active. When you are ready to continue receiving host transmissions, you may type RCV again (in local mode) without a filename, and EMT will use the same RCV file.

If you omit filename and no RCV file is active, EMT issues an error message. If you specify a new RCV file while another RCV file is active, RCV closes the active file, and prepares the new file to receive the transmission.

Use the CLOSE command to deactivate an RCV file.

#### TCTL {tctl commands}

Pass this command line to the Shell command TCTL to configure the SIO line. It is not necessary to specify the line number (-line), although you may if you wish to operate on a different line than the one you are using. The SPEED and SYNC commands have been superseded by this direct invocation of TCTL.

#### INTERM {CR|LF|CRLF|VAX|'hex'}

Select the input line terminator. The default is CRLF.

#### OUTTERM {CR|LF|CRLF|'hex'}

Select the output line terminator. The default is CR. EMT transmits the selected hexadecimal value as the terminator for each line.

**XMIT** *pathname* Prepare to transmit the DOMAIN file specified to the remote host. If you omit 'pathname', or if you specify a file that does not exist, EMT issues an error message. When you issue this command, EMT remains in local mode. EMT transmits the file when you press <F1>.

When EMT completes the transfer, it closes the file and returns to the previous mode. EMT does not send an end-of-file (EOF) signal to the remote host. If the host requires an EOF, enter remote mode and transmit it manually.

EMT can also receive commands from the host. If the host transmits the sequence:

HOST-COMMAND-CODE (EMT COMMAND STRING) LINE-TERMINATOR

EMT interprets the string as an EMT command. Use the EMT command CODE to define HOST-COMMAND-CODE.

Input Line Terminators	EMT Response
CRLF	Converts sequence to a line feed, ignoring any null characters that may separate the pair.
CR	Converts sequence to a line feed and ignores LFs.
LF	Interprets it as a line feed, and ignores CRs.
VAX	Interprets both CR and CR-LF as terminators and converts them to line feed.

EMT (EMULATE \_TERMINAL)

'hex'

Converts the given hexadecimal value to LF.

ENSUBS (ENTER\_ SUBSYSTEM) -- Enter a protected subsystem.

## FORMAT

ENSUBS subsystem\_name

ENSUBS is used to enter a protected subsystem at Shell command level. ENSUBS creates a new process in which to run the subsystem Shell.

Once in the subsystem, the SUBS command can be used to create new managers for the subsystem or to seal data objects so that only managers of the subsystem can operate on them. Also, subsystem managers can be debugged conveniently in this mode using DEBUG, and protected data objects can be examined. Note, however, that access to protected objects requires prior use of the SUBS -UP command.

## ARGUMENTS

subsystem\_name  
(required)

Specify name of subsystem to be entered. The Shell will search the directory /SYS/SUBSYS for the file specified.

## NOTES

- ENSUBS XXX just invokes the command /SYS/SUBSYS/XXX in a new process (unless the current process is already running in Subsystem XXX). The new process shares the window of the creating process, and is therefore subject to the same restrictions found when logging into a window. To avoid the limitations, a new window, containing a Shell running in the subsystem, can be created using the DM. Press the <CMD> key; next to the prompt, type:

Command: cp /sys/subsys/xxx

- The access control list on the file /SYS/SUBSYS/subsystem\_name determines who can enter the subsystem 'subsystem\_name': whoever has read and execute rights to it can enter the subsystem. Usually, this capability should be restricted to the creators of the subsystem or to the System Administrator.

**EOFF**

**EOFF -- Deactivate the Shell's -E flag.**

## **FORMAT**

### **EOFF**

**EOFF** disables variable evaluation. Variables are evaluated only inside variable expression delimiters, ((expression)); otherwise, the Shell treats the `^var_name` expressions as strings and they are not evaluated. To enable variable evaluation regardless of the context in which the variable appears, specify **EON**.

By default, **EOFF** is in effect when a Shell is invoked.

If **EOFF** is specified in a Shell script, it remains in effect until that Shell script exits, or until over-ridden by an **EON** in a nested Shell script. When a Shell script exits, the state of variable evaluation is returned to the state in effect just before the script was invoked.

**EOFF** requires no arguments or options.

**EON -- Activate the Shell's -E flag.**

## **FORMAT**

### **EON**

EON enables variable evaluation regardless of the context in which the variables appear. Normally, variables are evaluated only inside variable expression delimiters, ((expression)); otherwise, the Shell treats the `^var_name` expressions as strings and they are not evaluated.

By default, EOFF is in effect when a Shell is invoked.

If EON is turned on in a Shell script, it remains on until that Shell script exits, or until over-ridden by an EOFF in a nested Shell script. When a Shell script exits, the state of variable evaluation is returned to the state in effect just before the script was invoked.

EON requires no arguments or options.

EQS (EQUAL\_STRING)

EQS (EQUAL\_STRING) -- Compare strings for equality.

## FORMAT

EQS [string1 [string2]]

EQS compares strings for equality, and sets the abort severity level accordingly.

## ARGUMENTS

If no arguments are specified, EQS always returns TRUE.

### string1

(optional)

Specify text string to test. If this is the only string given (i.e., 'string2' is not specified), return TRUE if 'string1' is empty; otherwise return FALSE.

Default if omitted: return TRUE

### string2

(optional)

Specify text string to compare against 'string1'. EQS returns TRUE if the strings are equal, and FALSE if they are not.

Default if omitted: test 'string1' only

## EXAMPLES

The following Shell script will compile the PASCAL module named by the first argument (^1) if the second argument (^2) is '-c'. Then it will bind the module with 'library'.

```
if eqs ^2 '-c' then pas ^1 endif
bind ^1.bin library -b ^1
```

If the second argument is not '-c', or if there is no second argument, the program simply binds the module.

ESA (EXTERNAL\_SYMBOL\_ADDRESS) -- Display address of external symbol.

## FORMAT

**ESA symbol\_name**

ESA displays the address of an external symbol in an installed library. This command is primarily intended for system-level debugging.

In addition to displaying the address of an external symbol, ESA returns TRUE to the Shell if the symbol exists in an installed library and FALSE if it does not. This means that you may use ESA in conjunction with the Shell IF statement to determine whether or not a library is installed, provided you know the name of one of its symbols. For example, you might place the following lines in a Shell script:

```
if not esa my_favorite_symbol >/dev/null >?/dev/null then
    inlib my_library
endif
```

Note that in this instance, only the value returned by ESA is relevant; the actual address of the symbol does not matter. Hence all textual output is redirected into /dev/null.

## ARGUMENTS

**symbol\_name**  
(required)

Specify symbol whose address you wish to display. ESA is case sensitive with respect to the symbol name. Lowercase must be used to refer to symbols defined in FORTRAN and Pascal programs. Mixed case may be used, as needed, for symbols defined in C programs.

## EXAMPLES

```
$ esa gpr_$init
A1580C
$
```

This command displays the address of GPR\_\$INIT. This symbol resides within the GPR library, which was installed at system startup time.

## EXFLD (EXTRACT\_FIELDS)

**EXFLD (EXTRACT\_FIELDS) -- Manipulate fields of data.**

### FORMAT

**EXFLD {field\_spec} output\_format [pathname ...]**

EXFLD manipulates data kept in formatted fields. It copies data from specified fields of the input files to specified places in standard output.

### ARGUMENTS

**field\_spec**  
(required)

Specify either one of the following two arguments:

**field\_list**

Integer list identifying fields in the input file to be copied. Up to 9 input fields are allowed. You can specify a field by the columns in which it occurs or by its starting column and length. For example, 5-10 denotes a field that extends from column 5 through column 10, and 3+2 denotes a field that starts in column 3 and spans 2 columns. When specifying more than one field, separate the specifications with commas, for example:

5-10,16,72+8

Fields can overlap, and need not be in ascending numerical order. Thus

1-25,10,3

is a valid field specification.

**-T [c]**

Free-format separator specification. If input fields do not fall in certain columns, but rather are separated by some character (such as a blank or a comma), describe the fields by using '-T c', replacing 'c' with the appropriate separator. A tab character is the default for 'c'.

**output\_format**  
(required)

Specify literal string representing output format. Fields from input are referred to as \$1, \$2, \$3, and so forth, denoting the order in which the fields are specified. Up to 9 fields are allowed, plus the argument \$0 which refers to the whole line. Place the \$n symbol in the output format wherever the corresponding field should appear, surrounded by any characters desired. For example, an output format of:

"\$2 somewords \$1"

would produce an output line such as:

field2 somewords field1

**pathname**  
**(optional)**

Specify input file to be manipulated.

Default if omitted: read standard input

**EXAMPLES**

```
$ exfld 1-5,14-18 "$2 follows $1"  
ABCDE is not DEFGH  
DEFGH follows ABCDE  
*** EOF ***  
$
```

Specify extraction.  
Input text from standard input.  
Result.  
Signal completion with CTRL/Z.

## EXISTF

**EXISTF** -- Check for existence of an object.

### FORMAT

**EXISTF** *pathname* ...

**EXISTF** reads the object *pathname(s)* you supply and checks to see if the object exists. If the object does exist, **EXISTF** returns with a good program status (**PGM\_**\$TRUE). If the object does not exist, **EXISTF** returns an error status (**PGM\_**\$FALSE).

### ARGUMENTS

**pathname**  
(required)

Specify the object to be checked. Multiple pathnames and wildcarding are permitted. If you specify more than one *pathname*, all the objects must exist for **EXISTF** to return TRUE.

### EXAMPLES

```
1. $ if existf my_file then args "The file is there."      Test for "my_file"
   $_else args "Out of luck." endif                       which does not
   Out of luck.                                           exist.
   $
```

**EXISTVAR (EXIST\_VARIABLE) -- Check that a variable is set.**

**FORMAT**

**EXISTVAR var\_name ...**

The EXISTVAR command checks to see if the variable name(s) declared as its argument(s) has a currently set value. If the variable is currently set, EXISTVAR returns a "TRUE" value. If the variable is not currently set, EXISTVAR returns "FALSE". If you specify more than one variable name to check, all the variables must exist for EXISTVAR to return "TRUE".

**ARGUMENTS**

**var\_name [...]**  
**(required)**

Specify the variable name to be checked. Multiple names are permitted, separated by blanks.

## EXIT

**EXIT -- Exit from a loop.**

## FORMAT

### EXIT

EXIT terminates the flow of control in a Shell loop construct (FOR, SELECT, and WHILE). When EXIT is encountered, control passes to the first command following the body of the loop (see example below).

You may also interrupt the flow of control in a loop without actually leaving the loop by using the NEXT command. See the NEXT command description for more information.

Do not confuse this command with the DM command EX, which exits the Display Manger and returns control to the Boot Shell. See the EX command description in the DM commands chapter for more information.

The EXIT command requires no arguments or options.

## EXAMPLES

Consider the following section from a Shell script:

```
WHILE ((true))
DO   READC a
     IF ((^a = "y")) THEN EXIT ENDIF
     ARGS "still looking ..."
ENDDO
ARGS "Finished."
```

When the READC (READ\_CHARACTER) command reads a character into variable "a" that matches the character "y", the EXIT command executes and causes the script to jump to the command following the ENDDO.

For more information on variables, refer to the *DOMAIN System User's Guide*.

**EXPORT -- Change a Shell variable into an Environment variable.**

## FORMAT

**EXPORT var\_name...**

The Shell can access environment variables using all of the standard variable commands and operators. The EXPORT command adds the capability of turning regular Shell variables into environment variables.

Environment variables are variables that programs can access or set and that are used to store global state information. Several are generated automatically when you create a process; they can be displayed using the LVAR (LIST\_VARIABLES) command. For example:

```
$ lvar
environment NODETYPE = DN400
environment TZ = EST5EDT
environment PATH = ~com:/usr/ucb:/bin:/com:/usr/bin
environment TERM = apollo_15P
environment HOME = //node_8e4/joseph
environment USER = joseph
environment LOGNAME = joseph
environment PROJECT = none
environment ORGANIZATION = r_d
environment NODEID = 8E4
$
```

Environment variables are of special interest to users of DOMAIN/IX. Consult the DOMAIN/IX documentation for additional information.

**NOTE:** The Shell creates environment variables in UPPERCASE only. (Environment variables are case sensitive in DOMAIN/IX; the Shell only allows uppercase ones to avoid collisions between environment variables and Shell variables.)

## ARGUMENTS

**var\_name**  
(required)

Specify the Shell variable to be changed into an environment variable. It doesn't matter whether or not the name is typed in uppercase; the Shell converts it to uppercase automatically. Multiple variable names are permitted, separated by blanks. If the specified variable does not exist, EXPORT creates it.

## EXAMPLES

```
$ eon
$ current_dir := "//panacea/joe"
$ lvar
string current_dir = //panacea/joe      (Shell variable created.)
environment USER = joe
environment LOGNAME = joe
```

## EXPORT

```
environment PROJECT = none
environment ORGANIZATION = r_d
environment NODEID = D5B
environment PATH = :~com:/usr/ucb:/bin:/com:/usr/bin
environment TERM = apollo_19L
environment NODETYPE = DN300
environment TZ = EST5EDT
environment HOME = //panacea/joe
$ export current_dir
$ lvar
environment USER = joe
environment LOGNAME = joe
environment PROJECT = none
environment ORGANIZATION = r_d
environment NODEID = D5B
environment PATH = :~com:/usr/ucb:/bin:/com:/usr/bin
environment TERM = apollo_19L
environment NODETYPE = DN300
environment TZ = EST5EDT
environment HOME = //panacea/joe
environment CURRENT_DIR = //panacea/joe (Environment variable created.)
```

**FIND\_ORPHANS -- Locate and catalog uncataloged objects.**

## FORMAT

**FIND\_ORPHANS [options] [volume\_pathname]**

FIND\_ORPHANS finds all uncataloged permanent objects in a local volume. It uses or creates a directory ORPHANS in the root of the volume and enters the names of all objects not cataloged elsewhere. Uncataloged directories are found first, so no redundancy occurs.

The user of this command must have read permission to all directories on the volume. If some directory is not readable, every object under that directory will be cataloged in the ORPHANS directory. In addition, the user must either have permission to create the ORPHANS directory or to catalog objects in ORPHANS when it already exists.

The objects cataloged by FIND\_ORPHANS are given sequential names like F1, F2, etc., and can be moved using MVF to a directory of the user's choice.

This command is useful for finding objects that are lost by a broken directory. It should be run only on a quiescent node: i.e., one not connected to the network (use NETSVC -N to disable network communications) and not actively running any processes other than the one performing the FIND\_ORPHANS operation.

## ARGUMENTS

**volume\_pathname**  
(optional)

Specify the name of the volume to be searched. The volume must be physically attached to your node; you may not find orphan objects on volumes elsewhere in the network.

Default if omitted: search node boot volume

## OPTIONS

**-V[ERIFY]** Verify only; don't catalog any orphans

## EXAMPLES

```
$ find_orphans
11EE936C.50000105 -> f1
1216E28E.40000105 -> f2
12A2BC34.40000105 -> f3
12B782DC.40000105 -> f4
12B78321.50000105 -> f5
12B78353.60000105 -> f6
12B783EF.00000105 -> f7
12B784E3.90000105 -> f8
12B7863C.30000105 -> f9
12C18DBE.40000105 -> f10
12F98201.40000105 -> f11
13452895.80000105 -> f12
```

## FIND\_ORPHANS

```
140090B4.40000105 -> f13
140090F4.E0000105 -> f14
15322D3A.70000105 -> f15
17872C66.50000105 -> f16
Number of orphans: 16
$ ld /orphans -a
```

Directory "/orphans":

sys	type	blocks	current			
type	uid	used	length	attr	rights	name
file	rec	2	1462	P	p-ndwrx	f1
file	nil	0	0	P	p-ndwrx	f10
file	obj	1	58590	P	p-ndwrx	f11
file	nil	4	4096	P	p-ndwrx	f12
file	nil	4	4096	P	p-ndwrx	f13
file	nil	4	4096	P	p-ndwrx	f14
file	uasc	0	245	P	p-ndwrx	f15
file	mbx	17	280172	P	p-ndwrx	f16
file	nil	0	0	P	p-ndwrx	f2
file	nil	0	0	P	p-ndwrx	f3
file	obj	66	65862	P	p-ndwrx	f4
file	obj	134	135724	P	p-ndwrx	f5
file	rec	9	8412	P	p-ndwrx	f6
file	obj	115	116188	P	p-ndwrx	f7
file	mbx	15	278636	P	p-ndwrx	f8
file	nil	0	0	P	p-ndwrx	f9

16 entries, 371 blocks used.

**FLEN (FILE\_LENGTH) -- Count lines, words, and characters in a file.**

## FORMAT

**FLEN [options] [pathname ...]**

FLEN prints the number of lines, words, and characters in each of the named files. A word is defined as any sequence of characters delimited by tabs, spaces, and NEWLINEs. If more than one file is specified, totals for all the files are printed, also.

## ARGUMENTS

**pathname**  
**(required)**

Specify input file. Multiple file names and wildcarding are permitted.

Default if omitted: read standard input; suppress total counts

## OPTIONS

If no options are specified, all counts are reported.

**-L** Print only line counts.

**-W** Print only word counts.

**-C** Print only character counts.

Options may be mixed to achieve the desired reporting results.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ flen -L -C mary    Print the number of lines and characters in the
                    file 'mary'.
```

FMC (FORMAT\_MULTI\_COLUMN)

FMC (FORMAT\_MULTI\_COLUMN) -- Format text into multiple columns.

## FORMAT

**FMC** [**options**] [**pathname ...**]

FMC reads the named files and formats them into multiple columns on standard output. Each input line is placed in one column of an output line; input lines which are longer than the output column width are truncated. This command is useful to format text which is already in the form of a column or list.

## ARGUMENTS

**pathname**  
(optional)

Specify input file. Multiple pathnames are permitted, separated by blanks.

Default if omitted: read standard input

## OPTIONS

The options control output format. If no options are specified, the default output format is:

number of columns	2
page length	55
column width	60
gutter width	8

- C n** Specify n columns. Default: 2.
- L n** Specify page length in n lines. FMC produces output in pages, but does not place separators between the pages. Default: 55.
- W n** Specify column width in n characters. Input lines longer than n characters are truncated. Default: 60.
- G n** Specify gutter width in n spaces. The gutter is the space between columns. Default: 8.
- D n** Specify display terminal as output device. The column width is set to n characters and the page size is set to 24 lines. The number of columns and the gutter width are computed to maximize the amount of information on the screen. Default: 10.

## EXAMPLES

```
$ crefs sample | fmc -c 3 -w 22 -g 4
```

This command line first produces a cross-referenced list of all the symbols in the file SAMPLE, then formats the report in a 3-column list.

**FMT (FORMAT\_TEXT) -- Format a text file.**

## FORMAT

**FMT [pathname ...] [options]**

FMT is a general purpose text formatting program, allowing you to arrange output text according to formatting directives embedded in the input file or typed on standard input.

By default, formatted text is written to standard output. You may redirect it to a file with the **-OUT** option.

## ARGUMENTS

**pathname**  
(optional)

Specify input file to be formatted. This argument must precede any command line options. Multiple pathnames and wildcarding are permitted; however, FMT will concatenate multiple files prior to formatting. If FMT cannot find one of the specified input files, control shifts to standard input.

Default if omitted: read standard input

## OPTIONS

**-F n**

Begin output at the first page numbered n.

**-T n**

Terminate output at the first page numbered higher than n.

**-S**

Stop before printing each page, including the first. This option is useful for paper manipulation. The prompt "Type return to begin a page" is issued only once, before the first page.

**-PO n**

Page Offset. Shift the entire document n spaces to the right.

**-LF**

List names of files as they are processed.

**-OUT pathname**

Specify output file. If this option is omitted, formatted text is written to standard output.

## EXAMPLES

```
$ fmt mary -out mary.formatted -po 9      Format "mary" with a page offset
$                                           of 9 spaces, and write the
                                           results to "mary.formatted".
```

## Request Line Summary

Complete information on using FMT is available in the *DOMAIN System Utilities* manual.

# FMT (FORMAT\_TEXT)

Request	Initial	Default	Break	Meaning
.#			no	Ignore this line. Precede comment lines with this symbol.
.bd n		n=1	no	Boldface the next n lines
.bp n	n=1	n=n+1	yes	Begin new page and number it n
.br			yes	Break
.cc c	c=.	c=.	no	Control character becomes c
.ce n		n=1	yes	Center the next n input lines
.cu n		n=1	no	Continuously underline next n input lines
.de xx			no	Command xx; ends at .EN
.ef /l/c/r			no	Foots on even pages are l(left), c(enter), r(right). '#' and '%' produce page number and date, respectively.
.eh /l/c/r			no	Heads on even pages are l(left), c(enter), r(right). '#' and '%' produce page number and date, respectively.
.en			no	Terminate command definition
.fi	yes		yes	Begin filling output lines
.fo /l/c/r			no	Foot titles are l(left), c(enter), r(right) '#' and '%' produce page number and date, respectively.
.he /l/c/r			no	Head title is l(left), c(enter), r(right) '#' and '%' produce page number and date, respectively.
.in n	n=0	n=0	yes	Set left margin to column n+1
.ju	yes	yes	no	Begin justifying filled lines
.ls n	n=1	n=1	no	Set line spacing to n
.m1 n	n=3	n=3	no	Space between top of page and head
.m2 n	n=2	n=2	no	Space between head and text
.m3 n	n=2	n=2	no	Space between text and foot
.m4 n	n=3	n=3	no	Space between foot and bottom
.ne n		n=0	y/n	Need n lines; break if new page
.nf	no		yes	Stop filling
.nj	no		no	Stop justifying
.nr x m	m=0	m=0	no	Set number register x to m, -m, +m for decrement, increment
.of /l/c/r			no	Foots on odd pages are l(left), c(enter), r(right). '#' and '%' produce page number and date, respectively.
.oh /l/c/r			no	Heads on odd pages are l(left), c(enter), r(right). '#' and '%' produce page number and date, respectively.
.pl n	n=66	n=66	no	Set page length to n lines
.po n	n=0	n=0	no	Set page offset to n spaces
.rc c			no	Tab replacement character is c
.rm n	n=65	n=65	no	Set right margin to column n
.sc c	c=#	c=#	no	Change fixed space character to c
.so file			no	Switch input to file
.sp n		n=1	yes	Space n lines, except at top of page
.st n		n=0	yes	Space to line n from top; -n spaces to line n from bottom
.ta n1 n2 ...			no	Set tab stops at columns n1, n2, ...
.tc c			no	Tab character is c
.ti n		n=0	yes	Temporarily indent next output line n spaces
.ul n		n=1	no	Underline words in the next n input lines

## IN-LINE FLAGS

{_c_}	no	Underline characters enclosed in braces
{!c!}	no	Boldface characters enclosed in braces
@nc	no	Replace with value in number register c
#	no	Insert literal blank

## KEY

n denotes numerical values  
t denotes titles  
c denotes single characters

Signed numbers signify relative changes to a quantity; unsigned numbers signify absolute settings. Unless otherwise noted, omitted 'n' fields set the value to 1, omitted 't' fields are empty, and omitted 'c' fields restore the default character.

## FOR

**FOR -- Execute a FOR statement.**

## FORMAT

```
FOR var_name := int_expr [TO int_expr] [BY int_expr] command... ENDFOR
```

```
FOR var_name IN string_expr [BY {CHAR|WORD|LINE}] command... ENDFOR
```

FOR allows you to build a control structure that executes commands repeatedly as long as the result of a Boolean test is TRUE. The FOR command has two formats: one for assigning and testing integer expressions, and one for assigning and testing string expressions.

In the integer form, the (optional) TO and BY clauses permit you to specify ranges and increment values, respectively. For example, you might want to loop 5 times by saying

```
FOR a := 0 TO 10 BY 2
```

If you do not specify "BY int\_expr", the default increment is 1. If you do not specify "TO int\_expr", you will probably want to increment the variable manually inside the body of the loop. You should also put a test condition inside the loop (and probably use an EXIT to get out) or else you risk looping forever.

In the string form, the (optional) BY clause allows you to control the string assignment operation. If you specify "BY WORD" (the default), each word (a sequence of non-blank characters) in 'string\_expr' is assigned to 'var\_name' one at a time until 'string\_expr' is exhausted. You may also assign string values a character at a time, or a line at a time, by using the "BY CHAR" and "BY LINE" clauses, respectively.

## ARGUMENTS

<b>var_name</b> (required)	Specify the name of the Shell variable whose value is to be assigned and tested.
<b>int_expr</b> (required)	Specify any valid expression that returns an integer value.
<b>string_expr</b> (required)	Specify any valid expression that returns a string value.
<b>command...</b> (required)	Specify the command to be executed as long as the FOR test returns TRUE. This may be a Shell command, a Shell script, a variable assignment, or any other valid Shell operation. Multiple commands are permitted; separate them with semicolons or NEWLINE characters.

## EXAMPLES

1. The following example demonstrates the advantages of a FOR loop over a WHILE loop in one instance. Assume these line appear in a Shell script.

```
#
# A loop using WHILE.
#
eon
a := 0
while ((^a <= 10)) do
    args ^a
    a := ^a + 2
enddo
#
# The same loop using FOR.
#
FOR a := 0 TO 10 BY 2
    args ^a
ENDFOR
#
# End of script.
```

2. This example assigns three names to a variable.

```
#
# Script FILE_NAME
#
eon
FOR file IN "foo bar zap" BY word
    args ^file
ENDFOR
#
# End of script.
```

Execution produces:

```
$ file_name
foo
bar
zap
$
```

FPAT (FIND\_PATTERN)

FPAT (FIND\_PATTERN) -- Find a text pattern in an ASCII file.

## FORMAT

FPAT [options] [pathname... -P] reg\_expr ...

FPAT searches its input file(s) for lines matching the specified regular expressions and writes them to standard output or the file specified.

## ARGUMENTS

**reg\_expr...**  
(required)

One or more regular expression patterns. By default, a line that contains any of these expressions matches and is written to standard output. For a description of regular expressions used for pattern matching, see the chapter on DM basics. Patterns containing embedded spaces or Shell special characters must be enclosed in quotation marks.

**pathname -P**  
(optional)

Specify name of file to be searched. If you specify a pathname with this argument, you must follow it with "-P" to separate the pathname(s) from the search patterns on the command line. Multiple pathnames and wildcarding are permitted.

Default if omitted: read standard input

## OPTIONS

If no options are specified, any line that matches any one of the regular expressions is considered a matching line.

**-OUT pathname**

Write output to specified file. If input file names were specified, output filename can be derived. If this option is not specified, matching lines are written to standard output.

**-A**

Select only lines that match ALL regular expressions, in any order.

**-X**

Select only lines containing NONE of the regular expressions.

**-C**

Write only a count of matching lines, not the lines themselves.

**-I**

Ignore cases for search (i.e., become case-insensitive).

**-L**

Write line number with each line that matches the regular expression.

**-M n**

Set the maximum number of search lines to n (a decimal value). FPAT terminates after searching n lines.

- LF**                    Display the name of the file being examined before searching its lines.
- LM**                    Similar to -LF, but display the name(s) of only those file(s) which contain matches for the regular expression.
- RM n**                  Set maximum number of matches to be reported for this execution of FPAT.
- RMF n**                 Set maximum number of matches to be reported for each file being searched.

*Summary of Regular Expression Notation*

c	Literal character
?	Any character (except newline)
%	Beginning of line
\$	End of line
[...]	Character class (any one of these characters)
[^...]	Negated character class (all characters except those in brackets)
[c1-c2]	Any single character in the range c1 through c2
ec	Escaped character (e.g., @%, @[, @*)
en	Newline
et	Tab character
*	Closure (zero or more occurrences of previous pattern)
{...}	Tagged pattern

**EXAMPLES**

1. Assume the file "text" contains:

```
now
is
the
time
for
all
good
```

Then the command,

```
$ fpat text -p o                    produces ...
now
for
good
$
```

... and the command,

```
$ fpat -x -m 5 -l text -p o        produces ...
( 2) is
( 3) the
( 4) time
$
```

## FPAT (FIND\_PATTERN)

2. `$ fpat text?* -p the` Search for the string "the" in all files whose names begin with "text".
3. `$ fpat text?* -p the -out =.out` Search for the string "the" in all files whose names begin with "text", (i.e., "text", "text1", "text\_file", etc.) and write the output to the files "text.out", "text1.out", "text\_file.out", etc.

**FPATB (FIND\_PATTERN\_BLOCK) -- Find blocks of text containing patterns.**

## FORMAT

**FPATB [options] [pathname... -P] reg\_expr ... [-OUT pathname]**

FPATB reads blocks of text from its input files and writes them to its output file(s) so they meet the specified matching criteria. By default, blocks of lines are separated by an empty line or by a line containing only blanks. FPATB is similar to FPAT (FIND\_PATTERN) except that if a pattern is found, the entire block of lines is copied to output, rather than only the line in which the pattern occurs. Thus, it is useful for searching mailing lists, bibliographies, and similar files, where several lines are grouped together to form cohesive units.

## ARGUMENTS

**reg\_expr**  
(required)

Specify regular expression to be used for matching search. Each expression defines a text pattern, and you can specify up to nine expressions with each FPATB command. FPATB is case-sensitive; for example, "a" is different from "A". For a description of regular expressions used for pattern matching, see the chapter on DM basics.

**pathname -P**  
(optional)

Specify name of file to be searched. If you specify a pathname with this argument, you must follow it with "-P" to separate the pathname(s) from the search patterns on the command line. Multiple pathnames and wildcarding are permitted.

Default if omitted: read standard input

## OPTIONS

If no options are specified, any block containing a line that matches any one of the regular expressions is considered a matching block.

**-OUT pathname**

Write output to specified file. If input file names were specified, output filename can be derived. If this option is specified, it must be the last option on the command line (i.e., it must follow any regular expressions specified). If this option is not specified, matching lines are written to standard output.

**-A**

Select only blocks containing lines that match ALL regular expressions, in any order.

**-X**

Select only blocks containing NONE of the regular expressions.

**-C**

Write only a count of matching lines, not the lines themselves.

## FPATB (FIND\_PATTERN\_BLOCK)

- B reg\_expr1** Specify 'reg\_expr1' as the block separator, instead of a blank or empty line. Text blocks begin at lines containing 'reg\_expr1'. If -B is specified and -E is not, 'reg\_expr1' begins and ends the block.
- E reg\_expr2** Specify 'reg\_expr1' to start a block and 'reg\_expr2' to end a block. Note that the -E option is used only in conjunction with the -B option.
- L n** Write only the first n lines of selected blocks. If a block contains fewer than n lines, this option pads the output block with blank lines.
- LF** Display the name of the file being examined before searching its lines.

### *Summary of Regular Expression Notation*

c	Literal character
?	Any character (except newline)
%	Beginning of line
\$	End of line
[...]	Character class (any one of these characters)
[^...]	Negated character class (all characters except those in brackets)
[c1-c2]	Any single character in the range c1 through c2
@c	Escaped character (e.g., @%, @[, @*)
@n	Newline
@t	Tab character
*	Closure (zero or more occurrences of previous pattern)
{...}	Tagged pattern

### EXAMPLES

```
$ fpatb address_list -p 01824 -out zip_list   Locate text blocks with
$                                             the string "01824" in
                                             the file "address_list"
                                             and write the results
                                             to "zip_list".
```

FPPMASK (FLOATING\_POINT\_MASK) -- Set/display floating-point error mask.

## FORMAT

FPPMASK [options]

FPPMASK sets or displays the state of the floating-point package error mask for a process. The error mask specifies some of the conditions that constitute a floating-point exception for the process.

## OPTIONS

If no options are specified, the current floating-point error mask condition is displayed.

**-D condition ...**

Disable 'condition' (see below). Both conditions may be specified.

**-E condition ...**

Enable 'condition' (see below). Both conditions may be specified.

'condition' may be either of the following:

**LOS**

Loss of significance. This condition occurs when subtracting floating-point values that are exactly equal.

**UNDR**

Underflow. This condition occurs when a floating-point result is too small to be represented. For single-precision values, this error occurs at about 0.118E-37. For double-precision values, the error occurs at about 0.223E-307.

Both conditions are initially disabled when a process is created.

## EXAMPLES

1. `$ fppmask` Display current settings.  
     LOS (loss of significance): disabled  
     UNDR (underflow): disabled
2. `$ fppmask -e los undr` Enable both LOS and UNDR conditions.
3. `$ fppmask -d undr` Disable UNDR condition.

## FSERR (FIND\_SPELLING\_ERRORS)

**FSERR (FIND\_SPELLING\_ERRORS) -- Find spelling errors.**

### FORMAT

**FSERR [pathname ...] [options]**

FSERR copies the named files line-by-line to standard output, while looking up each word in a dictionary. If it finds any spelling errors on a line, or if it finds words that are not in the spelling dictionary, FSERR prints the line containing the questionable word and asks whether or not the word is spelled correctly. If you indicate that the word is misspelled, you are prompted for the correct spelling. FSERR corrects the spelling on standard output and continues.

FSERR uses three ASCII files. The large standard dictionary file is /SYS/DICT, which contains the bulk of the words known to FSERR. Add words to this file if you want them to become permanent additions to your dictionary, making sure entries remain in alphabetical order. (Use the SRF (SORT\_FILE) command to alphabetize the file if necessary.) If you do not wish to alter the standard dictionary, you may direct FSERR to use a file containing your own special words by specifying the -D option each time you invoke the command.

/SYS/DICTDX serves as an index into the large dictionary file to speed searches. Do not edit this file manually. If you make changes to /SYS/DICT, delete the index file; FSERR generates a new one if /SYS/DICTDX does not exist. Note that it takes some time to generate this index, so be prepared for a delay the first time you use FSERR after altering the dictionary.

Finally, a relatively few "common words" that occur with great frequency are stored in /SYS/CDICT. These are read and put into an internal hash table each time FSERR starts up, making access to them faster than looking in the large dictionary file. This list of words is **not** alphabetized; rather, words appear in order of relative frequency, with the most common words at the top of the file. You may make changes to this file if necessary. Just be careful not to make the file too big, since that would defeat the purpose of a high-speed lookup mechanism for common words.

### ARGUMENTS

**pathname**  
(optional)

Specify file containing text to be checked. Multiple pathnames are permitted separated by blanks.

Default if omitted: read standard input

### OPTIONS

- F** Process words just after a period ('.') in column 1 (i.e., FMT directives). The default is to ignore such "words".
- N** Process digits. The default is to ignore digits.
- U** Underline misspelled words instead of prompting for correction or verification.

FSERR (FIND\_SPELLING\_ERRORS)

- S** Collect and print statistics on dictionary use.
- C pathname** Write words that are not in the dictionary, but are correctly spelled, into 'pathname'.
- D pathname** Add the words in the file 'pathname' to the dictionary used for this run. Words in the file must appear one per line.

FST (FAULT\_STATUS)

FST (FAULT\_STATUS) -- Print fault status information.

## FORMAT

FST [options]

FST prints information about the most recent fault that occurred in the process. The information always includes the fault status, the program counter (PC) at which the fault occurred, and a textual description of the error (as reported by the system call `ERROR_$PRINT`).

This command is intended for system-level debugging.

If you are using a Peripheral Bus Unit (PBU) device, you can get fault information by using the option "-U" (see below).

## OPTIONS

- R Print the contents of the CPU general registers when the fault occurred.
- S Print the supervisor PC, entry control block (ECB), and status register (SR) if the fault occurred in supervisor mode. This option is ignored if the fault occurred in user mode.
- A Print all available fault information. (Prints the same information as both -S and -R.)
- U n Print the same information as both -S and -R for faults caused by the PBU interrupt handler for unit n.

## EXAMPLES

```
$ fst -a
Fault Diagnostic Information
Fault Status = 00120010:
process quit (from OS / fault handler)
User Fault PC = 000157FC
D0-D7: 00120010 00000000 00000002 FFFFFFFE 00000008 00000006 00000182 00000004
A0-A7: 0020A452 00E2F22E 0020A3D4 0020A450 00E2F174 0000C92C 002746B4 002746AC
Supervisor ECB = 00000000
Supervisor SR = 0000
Supervisor PC = 00000000
```

**HELP -- Provide help on Shell and DM commands.**

## FORMAT

**HELP [topic [subtopic] ]**

This feature provides information on Shell and DM commands and miscellaneous system services by opening a window to display the file that you request. For a list of subjects in the HELP library, type:

```
$ HELP INDEX
```

Access to system HELP files is also provided through the <HELP> key on low-profile keyboards. This key opens a read-only pad on a HELP file using your typed input to construct the pathname, so the syntax is slightly different if you are seeking information on a subtopic. In that case, separate the topic and subtopic with a slash (/) instead of a blank. For example:

```
Help on: shell/commands
```

## ARGUMENTS

**topic**

**(optional)**

Specify the name of the command or topic for which you desire help.

Default if omitted: display introductory information

**subtopic**

**(optional)**

Specify subtopic to be viewed. For example,

```
$ help shell commands
```

displays a topical index of Shell commands, while

```
$ help shell
```

displays general information about the Shell.

Default if omitted: no subtopic displayed

HLPVER (HELP \_VERSION)

HLPVER (HELP \_VERSION) -- Provide HELP support for Shell scripts.

## FORMAT

**HLPVER script\_name version ^1**

HLPVER provides access to the DOMAIN HELP system utilities that support the standard command options -HELP, -VERSION, and -USAGE for Shell commands. By placing the HLPVER command inside a Shell script, you can allow users of the script to specify these three standard command options and receive meaningful output.

HLPVER looks for help information in a file called /SYS/HELP/script\_name.HLP. HELP files have special information at the top that HLPVER uses. This information must follow a standard format. The following example shows the header of the HELP file for the WD (WORKING\_DIRECTORY) command.

```
1.1;wd (working_directory), revision 1.1, 81/06/27
WD (WORKING_DIRECTORY) -- Set or display the current working directory.
usage: WD [pathname]
```

All HLPVER output goes to standard output (normally directed to the process transcript pad). HLPVER returns the first line of the HELP file in response to -VERSION. It returns the second line through the first blank line in the file in response to -USAGE. It returns the entire file in response to -HELP.

Any user file placed in the /SYS/HELP directory is also available to the HELP command for display in a standard HELP window. Thus the file /SYS/HELP/MARY.HLP can be viewed with \$ HELP MARY regardless of whether or not you are using HLPVER inside the MARY script. HLPVER is solely for the purpose of enabling the three standard command options mentioned above.

## ARGUMENTS

**script\_name**

**(required)**

Specify the name of the script for which HELP is to be provided. The name is the right-most leaf in the pathname, not the entire pathname of the script. HLPVER uses this name to construct the pathname for the HELP file to be returned (i.e., /SYS/HELP/script\_name.HLP).

**version**

**(required)**

Specify the version number of the Shell script. HLPVER compares this number to the version number in the HELP file (the first characters in the file up to the first semicolon) and returns an error if they do not match. This allows you to coordinate versions of the script and the HELP file.

**^1**

**(required)**

Pass in the desired option from the command line. "^1" must appear literally so that HLPVER can tell what information to return (-HELP, -VERSION, or -USAGE). See the example below.

**EXAMPLES**

Assume that the following lines appear in a file called "test\_script":

```
#  
# Example script showing HLPVER usage.  
#  
hlpver test_script 1.0 ^1  
args "Please enter ..."
```

```
# End of script
```

When the user types:

```
$ test_script -help
```

HLPVER returns the contents of /SYS/HELP/TEST\_SCRIPT.HLP to the transcript pad. Likewise, when the user types:

```
$ test_script -version
```

HLPVER returns the first line of the HELP file containing the version number.

HPC (HISTOGRAM\_PROGRAM\_COUNTER)

HPC (HISTOGRAM\_PROGRAM\_COUNTER) -- Program counter histogram.

## FORMAT

**HPC [options] pathname [args...]**

HPC (HISTOGRAM\_PROGRAM\_COUNTER), part of DPAK (DOMAIN Performance Analysis Kit), looks at the performance of programs at the PC level.

HPC produces a histogram of the program counter during program execution, thus helping you locate the most compute-bound portions of your program.

While your program is executing, HPC samples the program counter at regular intervals, gathering a set of data points. Each data point records the region in which the program was executing -- the location of the program counter -- when the sample was taken.

HPC divides your program into 256 equally sized regions called "buckets." The size of the region depends on the size of your program or the range you select. The smaller the regions, the better the resolution of the analysis.

When execution of your program has ended, HPC displays statistics and a histogram (bar graph) of the program counter. Each bar corresponds to an area of program memory. The length of the bar indicates how much time the program spent executing in the corresponding area. HPC tells you which procedures and line numbers each bar represents.

While HPC and your program are executing, a serial line is not available for output.

## ARGUMENTS

**pathname**  
(required)

Specify the name of the program to be evaluated.

**args**  
(optional)

Specify any arguments to be passed to the program "pathname". These are not processed by HPC, but passed directly to your program.

Default if omitted: no arguments passed

## OPTIONS

If no options are specified, a histogram is produced for the entire program, with 500 samples taken per second.

**-LOW x**

Specify lowest address ('x') to be included in the histogram. 'x' must be a hexadecimal value. If this option is omitted, the histogram starts at the beginning of the program or procedure (see -FROM below).

**-HIGH x**

Specify highest address ('x') to be included in the histogram. 'x' must be a hexadecimal value. If this option is omitted, the

histogram continues to the end of the program or procedure (see -TO below).

**-FROM procedure**

Specify the beginning of a procedure as the lowest address to be included in the histogram. If both -FROM and -LOW are omitted, the histogram starts at the beginning of the program. Note the the procedure name is case-insensitive.

**-TO procedure**

Specify the end of a procedure as the highest address to be included in the histogram. If both -TO and -HIGH are omitted, the histogram stops at the end of the program. Note the the procedure name is case-insensitive.

**-PROC procedure**

Specify a single procedure to be included in the histogram. Note the the procedure name is case-insensitive.

By limiting the range of addresses in the histogram with -LOW, -HIGH, -FROM, -TO, and -PROC, you can study a specific part of your program, such as an I/O routine.

**-LIMIT n**

Limit the displayed histogram bars to those that represent more than 'n' % of the monitored program execution. The default value for 'n' is 1. Use -LIMIT 0 to show all histogram entries.

**-RATE n**

Specify how many times ('n') HPC samples the program counter per second. 'n' must be a decimal number in the range 5 to 2000. The default is 500 samples per second. A higher rate results in a more accurate histogram, but tends to slow program execution.

**-NHDR**

Generate the histogram without the header information. Using this option makes filtering the output easier.

**-MAP**

Generate a list of the names and starting and ending locations of the procedures in the program. This list is reduced if -FROM, -TO, -HIGH, or -LOW are used to restrict monitoring to specific procedures or memory addresses. The output from this option can be quite verbose for large programs.

**-BRIEF**

Produce a compact bar chart by showing only the name of the first procedure, or procedure fragment, contained in the bucket represented by each bar. By default, DPAT shows the names of all procedures or procedure fragments contained in the bucket. This option also suppresses source line information.

**EXAMPLES**

This section describes the use of HPC with the program VANDERBILT. First we call HPC, passing 'VANDERBILT' as an argument. Then we describe the output of HPC's analysis of VANDERBILT.

We invoke HPC as follows:

```
$ HPC VANDERBILT
```

# HPC (HISTOGRAM\_PROGRAM\_COUNTER)

Note that any HPC option would precede the name 'VANDERBILT'.

HPC displays the following (each of the note numbers, e.g., {1}, are explained after the output).

{1}

```
Address Size      Section
002D8040 00000384  PROCEDURE$
002A6250 000000A8  DATA$
002D83C4 0000015A  DEBUG$
```

{2}

VANDERBILT Done.

{3}

```
Program //corey/d_s/vanderbilt.bin from 002D8040 to 002D843F in 4-byte
increments
87178 interrupts, 7654 low misses, 2671 high misses, 67556 process
misses
9297 measurements in 209 buckets, 11% percent of total
```

{4}

Address	Proc. Name	Stmt. No.	% of Hits
002D811C	CONTRACTOR	[ 73 ]	1.2%  *
002D8120	CONTRACTOR	[ 75 ]	1.0%  *
002D8124	CONTRACTOR	[ 76 ]	1.5%  *
002D8128	CONTRACTOR	[ 76 ]	1.5%  *
002D812C	CONTRACTOR	[ 76 ]	1.2%  *
002D8230	DECORATOR	[ 100 ]	6.0%  *****
002D8234	DECORATOR	[ 100 ]	3.0%  ***
002D8238	DECORATOR	[ 100 ]	7.2%  *****
002D8308	ELECTRICIAN	-110 ]	2.7%  **
002D830C	ELECTRICIAN	[ 110 ]	1.3%  *
002D8310	ELECTRICIAN	[ 111 ]	1.5%  *
002D8314	ELECTRICIAN	[ 112 ]	1.7%  *
002D8318	ELECTRICIAN	[ 112 ]	1.4%  *
002D831C	ELECTRICIAN	[ 114 ]	13.7%  *****
002D8320	ELECTRICIAN	[ 114 ]	8.5%  *****
002D8324	ELECTRICIAN	[ 114 ]	16.6%  *****
002D8328	ELECTRICIAN	[ 116 ]	1.5%  *
002D832C	ELECTRICIAN	[ 116-	1.3%  *
002D8330	PLUMBER	-123 ]	2.5%  **
002D8338	PLUMBER	[ 124 ]	1.4%  *
002D833C	PLUMBER	[ 125 ]	1.2%  *
002D8340	PLUMBER	[ 125 ]	1.4%  *
002D8344	PLUMBER	[ 127 ]	4.2%  ****
002D8348	PLUMBER	[ 127 ]	2.5%  **
002D834C	PLUMBER	[ 127 ]	6.0%  *****
002D8350	PLUMBER	[ 129 ]	1.0%  *
002D8354	PLUMBER	[ 129-	1.3%  *
	Less than 1%		5.4%  *****

*Explanation of Notes*

- {1} A section map, showing the addresses and sizes of the program sections in VANDERBILT.
- {2} The output of the VANDERBILT program itself, which is simply the print statement 'VANDERBILT Done.'
- {3} HPC parameters and statistics are summarized as follows:

Program //corey/d\_s/vanderbilt.bin from (a) 002D8040 to  
 (b) 002D843F in (c) 4-byte increments  
 (d) 87178 interrupts, (e) 7654 low misses,  
 (f) 2671 high misses, (g) 67556 process misses  
 (h) 9297 measurements in (i) 209 buckets.  
 (j) 11% percent of total

Note that we have annotated the output to clarify the following descriptions.

from (a) to (b) The range of addresses in the sample. These addresses are derived from those specified with the -LOW and -HIGH or -FROM and -TO options after rounding to the nearest bucket address. If no option is specified, the range is the starting and ending addresses of the program.

*(c)-byte increments*

The bucket size derived from the equation  $(b-a)/256=c$  where  $b$  is the last address given, and  $a$  is the first address given in the "from (a) to (b)" range described above. Also, where 'c' is rounded up to the nearest power of 2. 256 is the maximum number of buckets.

(d) interrupts The number of times HPC sampled the program counter.

(e) low misses The number of data points below the Low address. This is specified with either the -LOW option, the -FROM option, or the low end of the program.

(f) high misses The number of data points above the High address. This is specified with either the -HIGH option, the -FROM option, or the high end of the program.

(g) process misses The number of data points ignored because they are not within the monitored process.

(h) measurements The number of real hits in the program. This number is derived from the equation  $h=d-(e+f+g)$ , that is, the number of interrupts minus the low, high, and process misses.

(i) buckets The number of buckets between the highest and lowest buckets in which there were any real hits, inclusive. HPC divides the range of addresses in the sample. Buckets are numbered consecutively from 0 according to increasing bucket



**IF** -- Execute a conditional statement.

## FORMAT

**IF** condition **THEN** command\_1 ... [**ELSE** command\_2 ...] **ENDIF**

IF executes a conditional statement depending on the results of a Boolean test. You can extend the IF command over several lines if you use it interactively or in a Shell script. When you use IF interactively, and extend the command over more than one line, the Shell prompts you for each new line of the command with the \$ \_ prompt (refer to the example below).

## ARGUMENTS

**condition**  
(required)

Specify a command or program to execute and test for truth, or specify a variable expression or Boolean variable to test for truth. "Truth" usually means that the command executes successfully (without any errors), or that the Shell variable expression or Boolean is "true". (Specifically, this argument is evaluated TRUE if it returns an abort severity level of 0 (zero).)

Refer to the *DOMAIN System User's Guide* for more information on Shell variables.

**command\_1**  
(required)

Specify command or program to execute if 'condition' returns TRUE.

**command\_2**  
(optional)

Specify command or program to execute if 'condition' returns FALSE (i.e., a severity level greater than zero).

## EXAMPLES

1. \$ IF eqs a a  
    \$ \_ THEN args "a is a"  
    \$ \_ ELSE args "Aristotle was wrong."  
    \$ \_ ENDIF  
    a is a  
    \$
2. IF eqs ^2 '-c'  
    THEN pas ^1  
        bind ^1.bin library -b ^1  
    ELSE bind ^1.bin library -b ^1  
    ENDIF

Example 2 might appear in a Shell script. These lines will compile the Pascal module named by the first argument (^1) if the second argument (^2) is '-c'. Then it will bind the module with 'library'. If the second argument is not '-c', or if there is no second argument, the command simply binds the module.

INLIB (INSTALL\_LIBRARY)

INLIB (INSTALL\_LIBRARY) -- Install a user-supplied library.

## FORMAT

**INLIB** *pathname*...

INLIB installs a library at the current Shell level; it remains installed until the Shell that installed it exits. (To load a library that is used by *all* processes, see note below.) The newly installed library will be used to resolve external references of programs (and libraries) loaded after its installation. (Thus, previously loaded libraries and programs will NOT be affected.)

Note that only those global references which have been MARKed by the binder become visible, and that the default action of the binder is to leave globals UNMARKed. You should, therefore, take care to MARK all appropriate globals when you bind your library. See the BIND command description for more details.

INLIB is an internal shell command.

**NOTE:** At Version 4.1 and later you can create a library that will be installed automatically in every process. This library resides in the file /LIB/USERLIB.PRIVATE. The procedure text in this library will be shared among all processes.

This library must be present at node startup time in order to be installed. After copying your library to /LIB/USERLIB.PRIVATE with the Shell command CPF (COPY\_FILE), you must shut down the node and start it up again in order to use the library. Changes to the library also require rebooting the node to load the new routines.

Global names in /LIB/USERLIB.PRIVATE must not duplicate names used in DOMAIN libraries.

## ARGUMENTS

**pathname**  
(required)

Specify name of library file(s) to be installed. Multiple pathnames and wildcarding are permitted.

## EXAMPLES

1. \$ inlib my\_lib

Install the library "my\_lib".

2. \$ inlib ?\*.lib

Install all files in the current working directory with a ".lib" suffix.

**INTM (INSTALL\_TYPE\_MANAGER) -- Install a type manager.**

## FORMAT

**INTM type\_name [mgr\_pathname] [options]**

INTM installs a new type manager for a specified type. The manager is copied into the type manager directory. This command does not accept wildcards.

## ARGUMENTS

<b>type_name</b> (required)	Specify the type for which the manager is to be installed.
<b>mgr_pathname</b> (optional)	Specify the pathname of the manager object file to install for this type.  Default if omitted: object file is named 'type_name'.

## OPTIONS

<b>-N node_spec</b>	Specify the node on which the type manager is to be installed. See the section on node specifications in Chapter 3 for more information. If this option is omitted, the type manager is installed on the current node.
<b>-L</b>	List the results of the operation.
<b>-R</b>	Replace an existing type manager if it exists.

## EXAMPLES

1. `$ intm example_type /mydir/my_example_mgr.bin`
2. `$ intm exmaple_type /mydir/old_example_mgr.bin -n //remote_vol -l "/mydir/old_example_mgr.bin" installed as the manager for type example_type on volume //remote_vol.`

INTY (INSTALL\_TYPE)

INTY (INSTALL\_TYPE) -- Install a new type.

## FORMAT

INTY type\_name source\_volume [-N node\_spec] [options]

INTY installs a type from one node to another. It will install both the type name and type manager on the target node (given by the -N option).

## ARGUMENTS

<b>type_name</b> (required)	Specify the name of the type to be installed.
<b>source_volume</b> (required)	Specify the pathname of the source volume from which to copy the type name and type manager.

## OPTIONS

<b>-N node_spec</b>	Specify the node on which the type is to be installed. See the section on node specifications in Chapter 3 for more information. You may also specify the entry directory of a volume mounted for software installation, as shown in the example below. If this option is omitted, the type is installed on the current node.
<b>-L</b>	List the results of the installation.
<b>-R</b>	Replace any existing type name/manager pair that currently exists.

## EXAMPLES

1. 

```
$ inty example_type //test_vol
Type "example_type" installed.
```
2. 

```
$ inty example_type //my_vol -n //test_vol -l
Type "example_type" installed on volume //test_vol.
```
3. 

```
$ mtvol w /mounted_disk
$ inty net_ethernet //rocket_j -n /mounted_vol -l
Type "net_ethernet" installed on //my_node/mounted_vol.
```

In this case, the disk has been mounted for software installation.

INVOL (INITIALIZE\_VOLUME) -- Initialize disk volumes.

**FORMAT**

**From AEGIS command Shell: INVOL**

**From Mnemonic Debugger : EX INVOL**

INVOL initializes physical disk volumes, creates logical volumes, and maintains badspot lists. Once initialized, a volume can be mounted with the MTVOL (MOUNT\_VOLUME) command, or can be used to bootstrap the operating system, providing it contains the necessary files.

For a detailed explanation of INVOL, see the *DOMAIN System Utilities* manual.

INVOL prompts for all required information.

## IOS\_TEST

IOS\_TEST -- Test IOS\_\$ calls

### FORMAT

#### IOS\_TEST [-INIT]

IOS\_TEST is a program for testing type managers that manage input and output to objects. IOS\_TEST allows you to open a stream to any type of object and then use selected IOS calls on the open stream. With IOS\_TEST, you can open streams to existing or new objects. For more information on using IOS\_TEST to test type managers, see *Using the Open System Toolkit for Extending the Streams Facility*. Complete descriptions of the IOS\_TEST interactive commands are available in the *DOMAIN System Utilities* manual.

### OPTIONS

**-INIT** Call the IOS\_\$INITIALIZE routine (within a type manager) at startup time.

**LAMF (LAMINATE\_FILE) -- Laminate files.**

## FORMAT

**LAMF [pathname... ] [-S string]**

LAMF laminates the named files to standard output. That is, it concatenates the first lines of all input files, sequentially, and writes the result to standard output; and so on for the next input lines. If the files contain different numbers of lines, null lines are used for the missing lines in the shorter files.

**NOTE:** To insert a NEWLINE character between lines, use the escape sequence, @n, as a string argument. (See Example 4, below.)

## ARGUMENTS

**pathname**  
(optional)

Specify name(s) of file(s) to be laminated to standard output. Multiple pathnames are permitted, separated by blanks.

Default if omitted: read standard input for input lines. Use a hyphen (-) to specify standard input in a list of file names.

## OPTIONS

**-S string**

Specify a string of text to be placed in each output line at the point it appears in the command argument list. 'String' may not exceed 300 characters. Strings containing embedded spaces must be in quotes (" ").

## EXAMPLES

1. \$ lamf mary fred                      Laminate files "mary" and "fred" and write results to standard output.
2. \$ lamf jan - george                  Laminate lines from "jan", standard input, and "george", in that order.
3. \$ lamf -S "A line from A: " a -S ", and from B: " b

would produce:

A line from A: first line from a, and from B: first line from b

Note that the text strings inserted are not bound in any way to the position of the pathname arguments: you may place strings wherever you please. Those strings that contain literal blanks must be enclosed in quotes, as above.

Escape sequences are valid in string arguments. For example:

## LAMF (LAMINATE\_FILE)

4. `$ lamf mary -S @n fred`

Insert a NEWLINE character between each line from mary and fred, thus interleaving the lines from the two files.

**LAS (LIST\_ADDRESS\_SPACE) -- List objects mapped into the address space.**

## FORMAT

**LAS [options]**

LAS produces a list of objects mapped into the address space. Information printed includes the virtual address range, the starting address within the object, and its pathname if available (in that order).

This command is most useful for system-level debugging.

## OPTIONS

If no options are specified, LAS lists the address space of the current process.

- ALL**                      List all address space, including that occupied by AEGIS.
- F[ROM] address**            Begin listing at the hexadecimal address specified.
- T[O] address**              End listing at the hexadecimal address specified.
- PROC[ESS] name**            List addresses for the process named. Use the PST (PROCESS\_STATUS) command to display the names of existing processes.

## EXAMPLES

1. \$ las

VA Range	Obj Start	Pathname
8000 - 17FFF	0	/sys/node_data/global_data
18000 - 2FFFF	0	/lib/pmlib
30000 - 37FFF	0	/lib/syslib.peb
38000 - 4FFFF	0	/lib/kslib
50000 - 57FFF	0	/lib/trait_type_lib
58000 - 67FFF	10000	/sys/node_data/global_data
68000 - 9FFFF	0	/lib/streams
A0000 - A7FFF	0	/lib/vfmt_streams
A8000 - BFFFF	0	/lib/error
C0000 - E7FFF	0	/lib/swtlib
E8000 - F7FFF	0	/lib/ftnlib
F8000 - FFFFF	0	/lib/pbulib
100000 - 127FFF	0	/lib/gprlib
128000 - 14FFFF	0	/lib/clib
150000 - 157FFF	0	/lib/lisp_initlib
158000 - 15FFFF	0	/sys/node_data/global_rws
160000 - 16FFFF	20000	/sys/node_data/global_data
170000 - 187FFF	0	/lib/shlib
188000 - 19FFFF	0	/lib/TFP

LAS (LIST \_ADDRESS \_SPACE)

```

1A0000 - 1BFFFF      0 /lib/dialoglib
1C0000 - 1C7FFF      0 /sys/node_data/ipc_data
1D0000 - 1D7FFF    30000 /sys/node_data/global_data
200000 - 2AFFFF      0 -- temporary file --
2B0000 - 2B7FFF      0 /sys/node_data/dm_mbx
2B8000 - 2BFFFF      0 /com/sh
2C0000 - 2C7FFF      0 -- temporary file --
2C8000 - 2CFFFF      0 /com/las
2D0000 - 2F7FFF    B0000 -- temporary file --
BC0000 - BCFFFF      0 /help_area/worksite
BD0000 - BDFFFF      0 /jtj

```

2944 KB mapped.

2. \$ las -from 188000

VA Range	Obj	Start	Pathname
188000 - 19FFFF	0		/lib/tpf
1A0000 - 1BFFFF	0		/lib/dialoglib
1C0000 - 1C7FFF	0		/sys/node_data/ipc_data
1D0000 - 1D7FFF	30000		/sys/node_data/global_data
200000 - 2AFFFF	0		-- temporary file --
2B0000 - 2B7FFF	0		/sys/node_data/dm_mbx
2B8000 - 2BFFFF	0		/com/sh
2C0000 - 2C7FFF	0		-- temporary file --
2C8000 - 2CFFFF	0		/com/las
2D0000 - 2F7FFF	B0000		-- temporary file --
BC0000 - BCFFFF	0		/help_area/worksite
BD0000 - BDFFFF	0		/jtj

1408 KB mapped.

3. \$ las -f 188000 -t 200000

VA Range	Obj	Start	Pathname
188000 - 19FFFF	0		/lib/tpf
1A0000 - 1BFFFF	0		/lib/dialoglib
1C0000 - 1C7FFF	0		/sys/node_data/ipc_data
1D0000 - 1D7FFF	30000		/sys/node_data/global_data

288 KB mapped.

**LBR (LIBRARIAN) -- Combine object modules into a library.**

## FORMAT

**LBR {-C | -UPD} library\_pathname [module\_pathname] [options] [-]**

The librarian manages libraries of object modules. It adds, removes, or replaces modules in the library. As input, you must provide the pathname of a library you want to create or update, followed by an optional list of object module pathnames and processing options. As output, the librarian produces a new or updated library file.

You can use LBR in two ways: by entering complete LBR command strings, or by using the "-" (hyphen) option to ask the librarian to prompt you for multiple strings of module\_pathname arguments and options. By using prompting you can perform several operations on object modules *in the same library file*, without entering LBR each time.

For a complete description of the librarian, see the *DOMAIN Binder and Librarian Reference*.

### *Prompting*

The optional hyphen at the end of the command line requests the librarian to begin prompting. The hyphen is valid only on the line containing the LBR command, and must be the last item on the line. Note that prompting is also requested if the command line contains only the LBR command.

If you request prompting, the librarian processes the arguments and options on the current command line, then displays an asterisk (\*) on standard output. In response to the asterisk, you can enter additional module\_pathname arguments and options. For example:

```
$LBR -C mylib.lib - <RET>
*file1.bin -DEL my_module <RET>
*file2.bin -L -REPL file3.bin
*<RETURN>
```

Prompting ends when you enter the -END switch or press <RETURN> in response to the asterisk. After prompting ends, the librarian finishes creating or updating the library file.

### *Comment Statements*

You can include comments to an LBR command during a prompting session or in a Shell script. Comments must be delimited by braces, as in this example:

```
$LBR -UPD plot.lib
*plot_line.bin { Add PLOT_LINE procedure to library }
*{ Generate library directory }
*-L
*-END
```

The librarian ignores any comments when it processes the command line.

## LBR (LIBRARIAN)

### *Librarian Errors*

If a problem occurs during LBR execution, the librarian displays a message on error output. The message indicates the nature and severity of the problem. Error-level messages are issued for fatal conditions, which prevent the librarian from creating or updating a library file. Warning-level messages are issued for conditions that do not prevent the librarian from producing a library file, but the file's contents may not be what you were expecting.

## ARGUMENTS

**-C[REATE] | -UPD[ATE] library\_pathname**  
(required)

The pathname of the library output file must be specified on the command line before you can specify any option that performs an operation on a library (such as adding to, extracting from, or reporting about a library). The -C (CREATE) or -UPD (UPDATE) option must be specified with the library pathname argument to indicate whether you want to create or update a library. Remember that only one library output file can be specified per execution of the librarian.

**module\_pathname**  
(optional)

Specify an object module to be added into the library. Multiple pathnames and wildcarding are permitted. If omitted, no new object modules are added to the library.

## OPTIONS

The following options instruct the librarian to perform various tasks. Note that some options apply directly to a library, while other options act on modules within the library. Default options are indicated by "(D)".

- DEL module** Remove an object module from the library. If a module of the given name cannot be found in the library, a warning is issued. Note that the librarian is case-sensitive to the name 'module'.
- EX module [-O pathname]** Extract the named module from the library. If the pathname modifier is specified with -O, the module will be copied to that pathname. Otherwise, the module is copied to a file having the same name as the module. Note that the librarian is case-sensitive to the name 'module'.
- L** List a directory of the library contents to standard output. This report shows the name of each module in the library, with a list of section information and global declarations and references for each module.
- MSGS (D)** Cause LBR to issue purely informational messages such as a summary of the number of errors and warnings that occurred.
- NMSGs** Cause LBR to suppress issuing purely informational messages.
- NSYS (D)** Do not list global variables which are defined in the system

library when generating a listing of global definitions and references with the -L option.

**-REPL pathname**

Replace, in the library, any modules found in the file specified by pathname. This option has an effect equivalent to first deleting all the modules found in pathname from the library, and then adding all the modules in pathname back into the library. The advantage gained by using -REPL is that you do not need to know the names of the modules in 'pathname'. Also, if you attempt to add a module to a library without using the -REPL option, and a module of that name *does* already exist, an error message is issued. If a module found in pathname *does not* already exist in the library, a warning message is issued.

**-SYS**

List global variables which are defined in the system library when generating a listing of global definitions and references with the -L option.

**- (hyphen alone)**

Request librarian prompting for further arguments.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

**EXAMPLES**

Refer to the *DOMAIN Binder and Librarian Reference* for detailed examples of LBR.

## LCNET (LIST\_CONNECTED\_NETWORKS)

LCNET (LIST\_CONNECTED\_NETWORKS) -- Display internet routing information.

### FORMAT

LCNET [options]

LCNET displays the list of known networks, their distances from the current node, the router used as the first hop towards that network, and other information.

The distances (hops) towards remote networks are measured in intervening routers. The distances are all for one-way traffic; if a network is three hops away from yours, your requests pass through three routers to get to that network. The responses also pass through three routers on the way back.

The -CONN option shows you the full internet topology, i.e. the list of networks and how the routers connect them together.

### OPTIONS

Default options are indicated by "(D)."

**-LOCAL (D)** Display the 'First Hop' and 'Hops' information for each network in the internet. The first hop is the node ID of a router on your network. It is the first router used in sending packets from your network to the target network. Other routers are also used if the target network is more than one hop away from your own.

**-FULL** Display information showing how up-to-date the routing table is (the 'Age' and 'Expiration' columns) in addition to the 'First hop' and 'Hops' information shown by the -LOCAL option. -FULL also lists inaccessible networks.

**-CONN** Show which routers are connected to each network, and which other networks those routers touch. This option displays the 'Touching' information.

**-HW** Display the type of hardware used for each of the networks (ring or IIC).

The -CONN and -HW options may take several seconds to execute if you have a large internet.

**-N node-spec** Print another node's view of the internet. The outputs produced by -LOCAL and -FULL vary from node to node; -N affects these outputs. The -N option does not affect the output produced by the -CONN or -HW options, since the hardware and connectivity do not depend on a node's position in the internet.

**-C** The -C option suppresses the title over each output column. It also fills every line of the "Network" column produced by the -CONN option, and every line of the "Hardware" column produced by the -HW option. These format changes make it easier to use LCNET's output as another program's input.

**EXAMPLES**

In this example, the node is directly connected to network COFFEE. Networks 5A1AD and ED1F1CE were connected in the past, but are not now accessible (perhaps because the routers are down).

The expiration date and time for the 'local' network is meaningless.

```
$ lcnet -full
      First
Network Hop Hops Age Expiration date/time
=====
      BO20 4B6F 1 NEW 1985/06/16 14:33:21
      BOOB00 4B6F 2 NEW 1985/06/16 14:33:21
      5A1AD 4B6F gone NEW 1985/06/16 14:33:21
      COFFEE 0 local NEW 1985/06/09 10:27:46
      ED1F1CE 4B6F gone NEW 1985/06/16 14:33:21
      DODO BAD1 1 NEW 1985/06/16 14:33:39
```

The 'Touching' information describes your internet completely. This example includes several kinds of information: - Network DEFACED has one router, node 2A3B. That router connects DEFACED to EFFACED. - Network FACEOFF contains two routers, 31DC and 1371. Those routers connect FACEOFF to COCOA and COFFEE, respectively.

```
$ lcnet -conn
      Touching Touching
Network Router Network
=====
      FOOD 5COB DECAF
      36CF COFFEE
      5A1AD 459B COFFEE
      45BE ED1F1CE
      BO02E 3FOA COFFEE
      COCOA BAD1 BOOB1E
      56B0 EFFACED
      31DC FACEOFF
      DECAF 5COB FOOD
      BOOB1E BAD1 COCOA
      COFFEE 36CF FOOD
      459B 5A1AD
      3FOA BO02E
      1371 FACEOFF
      DEFACED 2A3B EFFACED
      ED1F1CE 45BE 5A1AD
      EFFACED 56B0 COCOA
      2A3B DEFACED
      FACEOFF 31DC COCOA
      1371 COFFEE
```

```
$ lcnet -conn -c
      FOOD 5COB DECAF
      FOOD 36CF COFFEE
      5A1AD 459B COFFEE
      5A1AD 45BE ED1F1CE
      BO02E 3FOA COFFEE
      COCOA BAD1 BOOB1E
      COCOA 56B0 EFFACED
```

LCNET (LIST \_CONNECTED \_NETWORKS)

COCOA	31DC	FACEOFF
DECAF	5COB	FOOD
BOOB1E	BAD1	COCOA
COFFEE	36CF	FOOD
COFFEE	459B	5A1AD
COFFEE	3FOA	B002E
COFFEE	1371	FACEOFF
DEFACED	2A3B	EFFACED
ED1F1CE	45BE	5A1AD
EFFACED	56B0	COCOA
EFFACED	2A3B	DEFACED
FACEOFF	31DC	COCOA
FACEOFF	1371	COFFEE



LCNODE (LIST\_CONNECTED\_NODES)

LCNODE (LIST\_CONNECTED\_NODES) -- List nodes connected to the network.

## FORMAT

LCNODE [options]

LCNODE lists the nodes currently connected to the network. The list contains the ID of every node connected, the time at which the node was started, the current time, and the name of each node's entry directory.

This command reports only the nodes that respond within a preset time limit. Should a node be connected, but temporarily unable to respond within the specified time, it will not appear in the produced list.

## OPTIONS

- M[E]** Request information about your node only. This option displays the node ID.
- B[BRIEF]** Request brief output. LCNODE lists only the entry directory name for each connected node. Note that the entry directory of a diskless node is the entry directory of its paging partner.
- ID** When used with **-BRIEF**, display the node ID in addition to the entry directory.
- C[OUNT]** Request node count only. LCNODE lists only the number of nodes responding to its query.
- MAX[NODES] n** Set a limit on the number of nodes you want to see, even if more could have responded.
- FROM node\_spec** Starts the node list at some node other than your own. This is especially useful in an internet environment, for looking at networks other than your own. See the section on node specifications in Chapter 3 for more information.
- NAME** When you specify the **-BRIEF** option, LCNODE normally prints the entry directory for each node. If you specify **-NAME** with **-BRIEF**, LCNODE prints the node-name catalogued with the naming server. Only diskless nodes are printed differently. A diskless node's entry directory is its partner's node name; a diskless node's node-name is uniquely its own.

Unless the **-FROM** option specifies your own node, the list will only include an unbroken sequence of nodes running AEGIS SR9.0 or later. The rest of the node list is lost, starting with the first running a pre-SR9.0 AEGIS.

## EXAMPLES

## 1. \$ lcnode

The node ID of this node is 21.  
3 other nodes responded.

ID	Boot time	Current time	Entry Directory
21	1984/06/09 9:21:44	1984/06/09 16:06:22	//dollar
17	1984/06/09 13:52:02	1984/06/09 16:06:13	//quarter
27	1984/06/09 12:53:28	1984/06/09 16:06:07	//nickel
11	1984/06/09 12:03:39	1984/06/09 16:06:15	** DISKLESS ** //diskless_\$11 partner node: 17

## 2. \$ lcnode -me

The node ID of this node is 21.

## 3. \$ lcnode -b

```
//dollar
//quarter
//nickel
//quarter
```

(//QUARTER appears once as the host for a diskless node and once for the node with the disk.)

## 4. \$ lcnode -b -name

```
//dollar
//quarter
//nickel
//diskless_$000011
```

(-NAME shows you the name under which diskless node 11 is catalogued)

## 5. \$ lcnode -c

466 other nodes responded.

## 6. \$ lcnode -c -m

The node ID of this node is 116A.  
466 other nodes responded.

## 7. \$ lcnode -b -id

```
21 //dollar
17 //quarter
27 //nickel
11 //quarter
```

## 8. \$ lcnode -from OFAD.3924 -max 2

Starting from node 3924.  
1 other node responded,  
but more might have responded with a high -MAX value.

Node ID	Boot time	Current time	Entry Directory
3924	1985/02/14 17:20:45	1985/02/14 19:07:04	//laurel
34Bf	1985/02/14 18:46:52	1985/02/14 19:08:09	//hardy

LD (LIST\_DIRECTORY)

LD (LIST\_DIRECTORY) -- List contents of a directory.

## FORMAT

LD [pathname...] [options]

LD lists the objects in a directory on standard output. It provides a wide variety of information on the contents of the various objects, depending on the command options that you select.

## ARGUMENTS

**pathname**  
(optional)

Specify pathname of the object to be described. The object may be a directory, a file, or a link. If you specify a directory, LD describes the files in that directory. If you specify a file, the attributes of that file are reported. Multiple pathnames and wildcarding are permitted. (If they are used, each name is assumed to be a filename.)

Default if omitted: list contents of working directory

## OPTIONS

Default options are indicated by "(D)."

### *Attributes*

- A Display all attributes.
- ATTR Display permanent/immutable/trouble flags.
- BL Display disk blocks used.
- LEN Display current length in bytes.
- R Display your access rights to entries.
- ROOT Display the contents of the replicated root directory managed by the naming server helper.
- ST Display system object type.
- TU Display type UIDs.

### *Date and Time*

- D Display creation, modified, and last used dates.
- DTC Display date/time created.
- DTM Display date/time last modified.

**-DTU** Display date/time last used.

*Streams*

**-SI** Display all stream header information.

**-AB** Display streams ASCII/binary flag.

**-CONC** Display streams object concurrency.

**-RT** Display streams record type.

*Entry Selection*

**-CRB d** Display entries created before date and time "d".

**-CRA d** Display entries created after date and time "d".

**-USB d** Display entries used before date and time "d".

**-USA d** Display entries used after date and time "d".

**-MOB d** Display entries modified before date and time "d". Same as old **-BE** option.

**-MOA d** Display entries modified after date and time "d". Same as old **-AF** option.

**-BE d** Display entries modified before date and time "d". Obsolete option: use **-MOB**.

**-AF d** Display entries modified after date and time "d". Obsolete option: use **-MOA**.

**-DI** Treat all names as directory names and list the contents of those directories.

**-ENT** List attributes of the target object itself. This option has no effect if the pathname refers to a file. If the target object is a directory, **-ENT** causes LD to display attributes of the directory itself rather than its contents. If the target object is a link, **-ENT** causes LD to display attributes of the link itself rather than trying to resolve the link and display attributes of the resolution object. See Example 5 below.

**-LD** (D) List directory names. If this option is specified, then **-LF**, **-LL**, and **-LN** lose their default status, and must be specified explicitly, if desired.

**-LF** (D) List file names. If this option is specified, then **-LD**, **-LL**, and **-LN** lose their default status, and must be specified explicitly, if desired.

**-LL** (D) List link names. If this option is specified, then **-LD**, **-LF**, and **-LN** lose their default status, and must be specified explicitly, if desired.

## LD (LIST\_DIRECTORY)

**-LN** (D) List diskless node names. If this option is specified, then **-LD**, **-LF**, and **-LL** lose their default status, and must be specified explicitly, if desired. Diskless node names normally appear only when you specify **-ROOT**, or when you list the **//** directory.

**-LT** Display link resolution names.

### Output Control

**-SC** Sort the output vertically in columns.

**-SR** (D) Sort the output horizontally in rows.

**-W n** Adjust the output to be 'n' characters wide. If this option is omitted, LD automatically adjusts the width of the output to the size of the transcript pad's window, unless the command is issued from a dumb terminal or some other windowless device. In that case, the output defaults to 80 characters wide if **-W** is omitted.

**-C** List entries in a single column, suppress header.

**-HD** (D) Display header and totals.

**-NHD** Suppress header and totals.

**-SN** (D) Sort entries by name.

**-NSN** Suppress entry sorting.

**-WARN** (D) Produce a warning if no wildcard matches are found.

**-NWARN** Suppress warning if no wildcard matches are found.

LD uses the command line parser, and so also accepts the standard command options with the exception of the query options (**-QA**, **-NQ**, **-QW**).

### TIME

The time at which a file is created, modified, or used is accurate within a certain tolerance. The reported time of creation or modification is correct within one minute of the actual creation or modification time. The time of last use is updated only if more than an hour has elapsed since the recorded time of last use. Hence, the time of last use reported by the LD command may vary by as much as an hour from the actual time of last use.

### EXAMPLES

1. `$ ld -a`

Directory `"/col/users/final1"`:

sys	type	blocks	current				
type	uid	used	length	attr	rights	name	
file	rec	18	17640	P	pndwrx	ch1	
file	rec	18	18428	P	pndwrx	ch2	

```
file rec      67   67210 P   pndwrx  ch3
file rec      12   11554 P   pndwrx  ch4
```

4 entries, 115 blocks used.

2. \$ ld -dtm

Directory "/col/users/final1":

```
date/time
modified      name

82/03/28 17:18 ch1
82/03/28 17:18 ch2
82/03/28 17:19 ch3
82/03/28 17:20 ch4
```

4 entries, 115 blocks used.

3. \$ ld /sys/ins/[a-e]?\*.ins.ftn -a

```
sys  type  blocks  current
type uid    used   length attr rights  name

file rec      1     872 P   pndwrx  /sys/ins/base.ins.ftn
file rec      2    1274 P   pndwrx  /sys/ins/cal.ins.ftn
file uasc     20   19966 P   pndwrx  /sys/ins/core.ins.ftn
file rec      1     738 P   pndwrx  /sys/ins/ec2.ins.ftn
```

4 entries listed, 24 blocks used.

4. \$ ld //v?\* -a

```
sys  type  blocks  current
type uid    used   length attr rights  name

node                                     //victor
sdir nil          5    5120 P   -----rse //visitor
(attributes unavailable)                //void
sdir nil          3    3072 P   pgn-calrse  //vulture
```

4 entries listed, 8 blocks used.

NOTE: in this example, //victor is the name of a diskless node.

5. \$ crl foo //behemoth/rkd/foo.dat

```
$ ld foo -ll -lt
?(ld)  "foo" - name not found (OS/naming server)
```

{This error occurs because the resolution object //behemoth/rkd/foo.dat does not exist. Now use -ENT to show attributes of the link itself without trying to resolve it.}

```
$ ld foo -ll -lt -ent
```

```
foo  "//behemoth/rkd/foo.dat"
```

1 entry listed.

## LD (LIST\_DIRECTORY)

{The following command displays the contents of the working directory.}

```
$ ld . -a
```

Directory "//otis/tstlib/trash":

sys	type	blocks	current			
type	uid	used	length	attr	rights	name
file	uasc	1	32	P	pgndwrx	abc
link						foo

2 entries, 1 block used.

{Now display attributes of the working directory itself.}

```
$ ld . -ent -a
```

sys	type	blocks	current			
type	uid	used	length	attr	rights	name
dir	nil	2	2048	P	pgndcalrse	

1 entry listed, 2 blocks used.

**LKOB (LOCK\_OBJECT) -- Lock an object.**

## FORMAT

**LKOB** *pathname* [*options*]

LKOB locks the specified object. The locking constraint is "n readers XOR 1 writer".

LKOB is primarily used for system-level debugging.

To list locked objects, use LLKOB (LIST\_LOCKED\_OBJECTS). To unlock an object, use ULKOB (UNLOCK\_OBJECT).

## ARGUMENTS

<b>pathname</b> (required)	Specify object to be locked. Multiple pathnames and wildcarding are permitted.
-------------------------------	--

## OPTIONS

Default options are indicated by "(D)."

<b>-R</b>	<b>(D)</b>	Lock the object for reading.
<b>-W</b>		Lock the object for writing.
<b>-I</b>		Lock the object for reading, with intent to write.
<b>-R2W</b>		Change the lock mode of the object from "read" or "read-intend-write" to "write".
<b>-R2RIW</b>		Change the lock mode of the object from "read" to "read-intend-write".
<b>-W2R</b>		Change the lock mode of the object from "write" to "read".
<b>-W2RIW</b>		Change the lock mode of the object from "write" to "read-intend-write".

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ lkob susan -w           Lock file "susan" for writing.
```

## LLKOB (LIST\_LOCKED\_OBJECTS)

**LLKOB (LIST\_LOCKED\_OBJECTS) -- List locked objects.**

### FORMAT

#### LLKOB [options]

This command lists the locked objects resident on volumes mounted on this node, and objects resident in other nodes that are locked by processes running locally.

The listing for each object includes the locking constraints imposed on the object (e.g., n-readers XOR 1-writer), the specific lock mode being used (e.g., Read, Write, Read-Intending-Write), the network node ID of the node at which the object is located, the node ID of the node in which the locking process is active, and the name (if it is available) of the object itself.

### OPTIONS

**-R[EMOTE]** Specify list of only those objects that either reside on this node and are locked by another node, or reside on another node and are locked by this node (i.e., those objects whose locks are in some way remote).

**-C[OUNT]** List only a one-line summary of the number of objects locked.

### EXAMPLES

1. \$ llkob

USE	CONSTRAINT	HOME NODE	LOCKING NODE	FILE
W	nR_xor_1W	21	21	/sys/dm/pdb
R	nR_xor_1W	21	21	/sys/dm/fonts/std
W	nR_xor_1W	21	21	--Temporary File--
R	nR_xor_1W	21	21	--Uncataloged Permanent File--
W	nR_xor_1W	21	21	--Display Manager Pad--

2. \$ llkob -c  
locked: 102 -- 100 local, 2 remote; 100 locally locked, 2 remotely

**LOGIN -- Log in to a running process.**

## FORMAT

**LOGIN [person[.project[.org]] [-LP [passwd]] [-C pathname args... ]**

The LOGIN command allows you to log in to a running process with a different identity. This permits you to have multiple concurrent processes running under different Subject Identifiers (SIDs). See the EDACL command description for an explanation of SIDs.

As of SR9.2, the DM inherits the working and naming directories, plus the environment variables, of the new login process. This means that normal DM activities (like editing files) are now transparent to login activity. In releases prior to SR9.2, the DM did not assume the process identity and could not interact transparently with other process activities.

To log out of a running process and return control of the process to the original SID, enter an end-of-file mark (usually CTRL/Z) in the process input pad.

## ARGUMENTS

**person**  
(optional)

Specify your username. If you omit 'person', then LOGIN prompts you to log in interactively. In this case, respond just as you do to the DM log in prompt, i.e., issue the L command in the form:

L person[.project[.org]] [-P] [-H]

You will be asked for a password, and may take the opportunity to change your password and login home directory with the -P and -H options. See the L command description in the DM command chapter for more information. When you have successfully logged in, the process in the window assumes the new identity, and the node's local registry is updated.

Default if omitted: prompt for 'L' command

**project**  
(optional)

Specify project ID (if one exists). This ID may be separated from 'person' by either a blank space or a period.

Default if omitted: no project ID specified

**org**  
(optional)

Specify organization ID (if one exists). This ID may be separated from 'person' by either a blank space or a period.

Default if omitted: no organization ID specified

## LOGIN

### OPTIONS

**-LP [passwd]** Specify password. If the 'person' argument appears on the command line, and -LP is not specified, you will be prompted for a password. If -LP is specified without an associated password, a blank password is used. -LP must follow the 'person' argument and precede the -C option. In addition, "-C" cannot be the password. Note that using -LP makes your password visible in the window.

After a successful login, the node local registry is updated with the new identity.

**-C pathname [args ...]** Specify a program (followed by optional arguments to be passed to the program) that is to be invoked in the window after a successful login. If -C is not specified, /COM/SH (the Shell) is invoked. -C must not precede the 'person' argument.

### EXAMPLES

```
$ login
Please log in: 1 user
Password:
Logged in as user.none.none Monday, March 5, 1984 11:06:55 (EST).
$ args "And now for something completely different."
.
.
$ (CTRL/Z)
*** EOF ***
process stop (OS/fault handler)
$ # Control returned to original SID
```

**LOPSTR (LIST\_OPEN\_STREAMS) -- List open streams.**

**FORMAT**

**LOPSTR**

LOPSTR lists the streams that are open for the current process. The list contains the stream ID and access mode (read, write, append, and so forth) for each stream. The pathname (if one exists) associated with each stream is also displayed.

LOPSTR requires no arguments or options.

**EXAMPLES**

```
$ lopstr
st#  open  name
0 read   (standard input)
1 append (standard output)
2 read   (error input)
3 append (error output)

4 streams open.
```

LRGY (LIST\_REGISTRY)

LRGY (LIST\_REGISTRY) -- List registry sites.

## FORMAT

LRGY [options]

LRGY lists registry site names and the name of the network master registry file. For complete information on the use of local and network registries, see *Administering Your DOMAIN System*.

## OPTIONS

If no options are specified, LRGY lists the sites in the current node's registry file copy (/REGISTRY/REGISTRY).

**-R pathname** Specify name of registry file to be listed. If the -LOC option is also present, this name must be a node name (see example 3). If you omit this option but include -LOC, LRGY lists your node's local registry (/'node'/REGISTRY/LOCAL\_REGISTRY).

**-LOC** List local registry.

## EXAMPLES

1. \$ lrgy List registry file copy.  
Registry: //os/registry/rgy\_master  
Sites of registration data files:  
//os/registry/rgy\_dir.1  
//us/registry/rgy\_dir.2  
//tape/registry/rgy\_dir.3
2. \$ lrgy -loc List current node's local registry.  
Registry: //tape/registry/local\_registry  
Sites of registration data files:  
//tape/registry/local\_site  
Registry is LOCAL. It has 11 slots for login;  
the expiration period is 10 days.
3. \$ lrgy -r //os -loc List local registry of node "os".  
Registry: //os/registry/local\_registry  
Sites of registration data files:  
//os/registry/local\_site  
Registry is LOCAL. It has 11 slots for login;  
the expiration period is 10 days.

**LTY (LIST\_TYPES) -- List installed types.**

**FORMAT**

**LTY [options]**

LTY lists the types currently installed on a volume. It can also be used to list the contents of internal caches for debugging purposes.

**OPTIONS**

If no options are specified, LTY lists types installed on the boot volume.

**-N node\_spec**

Specify the node whose type names are to be listed. See the section on node specifications in Chapter 3 for more information. You may also specify the entry directory of a volume mounted for software installation, as shown in the example below.

**-U**

Display type UIDs as well as type names.

**-GLOB**

Display contents of global type name cache instead of the type file (for debugging only).

**-PRIV**

Display the contents of the private (per-user) type name cache instead of the type file (for debugging only).

**EXAMPLES**

```
$ lty
```

```
Local type file
```

```
area    bitmap  boot    casehm  ddf     evetype  hdru    ipad
lheap   mbx      mt      nil     null    obj      objlib  pad
pipe    rec     sch     sio     uasc    und
```

In the following example, the disk has been mounted for software installation. The disk's top level directory (catalogued as '/mounted\_disk' by the MTVOL command) must contain a "sys" directory. If it does not, you will get a "types file not found" error.

```
$ mtvol w /mounted_disk
```

```
$ lty -n /mounted_disk
```

```
Type file for "//my_node/mounted_disk"
```

```
area    bitmap  boot    casehm  ddf     evetype  hdru    ipad
lheap   mbx      mt      nil     null    obj      objlib  pad
pipe    rec     sch     sio     uasc    und
```

LUSR (LIST\_USER)

LUSR (LIST\_USER) -- List logged on users.

## FORMAT

LUSR [options]

LUSR lists the identities of active users on the network.

## OPTIONS

If no options are specified, the person name and node entry directory of all users logged into the DM are listed.

- ME** List the user logged on to this node by person, project, organization name, and node ID.
- N node\_spec ...** Lists user(s) logged on to the node specified. See the section on node specifications in Chapter 3 for more information. Multiple pathnames or node IDs are permitted; separate them with blanks.
- BR** Suppress listing of home directory names. Home directory names are listed if this option is not specified.
- FULL** List complete PPON (person, project, organization name, and node ID) for each user listed.
- NOFULL** List only the person name of each user listed. This is the default setting unless **-ME** is specified, in which case the full PPON is listed by default.
- ALLP** List identities for all user processes, not just the DM, either by node (if **-N** is also specified), by name (**-PPO**), for the current node only (**-ME**), or everywhere in the network.
- PPO ppo** List user(s) named, at all nodes to which they have logged in to the DM. 'ppo' is a string of the form 'pers.proj.org', where '%' may be used as a wildcard specifier and trailing %'s may be omitted (e.g., %os\_dev or joe.%r\_d).
- IDLE** Include idle nodes in report. If this option is omitted, LUSR suppresses the names of nodes at which no one is logged in.

## EXAMPLES

```
1. $ lusr -me
    loc.none.mfg.1D5 //ET
    $
```

```

2. $ lusr -me -nofull -br
    loc
    $

3. $ lusr -n //magic //mountain //park
    joe           //MAGIC
    brian         //MOUNTAIN
    gordon        //PARK
    $

4. $ lusr -full
    jtj.none.none.532      //zoid
    andy.none.now.12B     //me
    carol.none.mtg.334    //vip
    nelson.none.pres.838  *** diskless 383 ***
                          //halfwit partner node: //plan
    annie.none.r_d.6CA    //lunar
    now.system.advent.368 *** diskless 368 ***
                          //diskless_$000368 partner node: //zoid
    beth.none.mfg.2F7     //mack
    $

5. $ lusr -idle
    joe           //magic
    *No one logged in* //strider
    *No one logged in* //panacea
    janet        *** diskless //cutie *** partner node: //nirvana
    john         //duke
    eric         //lion
    *No one logged in* //fourbits
    harper       //basil
    $

```

## LVAR

**LVAR (LIST\_VARIABLES) -- List information about set variables.**

### FORMAT

**LVAR [var\_name ...]**

The LVAR command lists the type, name, and value of currently set variables. Optionally, you can specify individual variable names.

### ARGUMENTS

**var\_name ...**

**(optional)**

List type, name, and value of the specified variable(s).

Default if omitted: list information for all variables currently set

**LVOLFS (LIST\_VOLUME\_FREE\_SPACE) -- List free space on logical volumes.**

## FORMAT

**LVOLFS [pathname] [options]**

LVOLFS prints information about the amount of available storage on mounted volumes. This information includes the total amount of storage in disk blocks, the amount of free storage, the percent of the total storage that is free, and the entry directory name for the volume.

## ARGUMENTS

**pathname**  
(optional)

Report on the volumes mounted on the home node of the specified file.

Default if omitted: list free space on current node

## OPTIONS

If no options are specified, LVOLFS reports the storage available on the volumes mounted on the current node.

**-A**

Report on all volumes mounted in the network.

**-N node\_spec ...**

Report on the volumes mounted on the specified node[s]. See the section on node specifications in Chapter 3 for more information. Multiple 'node\_spec' strings are permitted; separate them with blanks.

## EXAMPLES

```
$ LVOLFS -A
```

# free	# total	% free	node id	entry directory
24217	30012	81	1A	/
16589	30012	55	2B	//DEV
7927	30012	26	3C	//LANG
14497	30012	48	4D	//MKT

## MACRO

**MACRO -- Expand macro definitions.**

### FORMAT

**MACRO [-0] [pathname ...]**

MACRO is a general-purpose macro processor. MACRO reads the files and writes to standard output a new file with the macro definitions deleted and the macro references expanded.

### ARGUMENTS

**pathname**  
**(optional)**

Specify file containing macro definitions to be processed. Multiple pathnames are permitted.

Default if omitted: read standard input

### OPTIONS

**-0**

(Zero, not letter O) Remove one level of brackets in macro calls prior to processing. Normally, brackets appearing outside any macro calls (level zero brackets) are NOT removed.

A macro is a symbolic constant; when you use MACRO, each constant is replaced by the string of characters which define it. The general form of a macro definition is:

```
DEFINE(name, replacement text)
```

The string 'name' can consist of letters (a-z and A-Z), digits (0-9), underscores (\_), and dollar signs (\$). All subsequent occurrences of the string 'name' separated from other letters, digits, underscores, and dollar signs by any other characters, spaces, or newline characters will be replaced by the replacement text. No space is allowed between the command (in this case, DEFINE), and the left parenthesis.

Blanks in definitions are significant; they should appear in the replacement text only where desired. Uppercase and lowercase letters are also significant. The replacement text may be more than one line long. However, when an entire macro definition is followed immediately by a newline, the newline is discarded. This prevents extraneous blank lines from appearing in the output.

A simple example of a macro is:

```
DEFINE(EOF, -1)
```

Thereafter, all occurrences of 'EOF' in the file would be replaced by '-1'.

You may specify arguments in macro definitions with the characters '\$n', where n is a number between 0 and 9. The arguments to be inserted when the macro is encountered are given inside parentheses following the macro name. \$0 refers to the name of the macro itself. For example,

```
DEFINE(copen,$3 = open($1,$2) )
```

defines a macro that, when called by

```
copen(name, READ, fd)
```

expands into

```
fd = open(name,READ)
```

If a macro definition refers to an argument that was not supplied, the \$n will be ignored. The \$ is taken literally if a character other than a digit follows it.

Macros can be nested, and can be called recursively. Any macros encountered during argument collection are expanded immediately, unless they are surrounded by square brackets ([ ]). That is, input surrounded by brackets is left absolutely alone, except that one level of [ and ] is stripped off. Thus it is possible to write the macro D as

```
DEFINE(D, [define($1,$2)])
```

The replacement text for D, protected by the brackets, is literally 'DEFINE(\$1,\$2)' so you could use:

```
D(a, bc)
```

to define a as bc. Brackets must also be used to redefine a macro. For example:

```
DEFINE(x, y)
```

```
.
```

```
DEFINE(x, z)
```

will define y in the second line, instead of redefining x. To define x the second time, the operation must be expressed as

```
DEFINE(x, y)
```

```
.
```

```
DEFINE([x], z)
```

Normally, brackets appearing outside any macro calls (level zero brackets) are *not* removed. When the -0 (zero, not letter O) option is specified, one level of brackets is removed both inside and outside the macros. One level of brackets is also removed when the macro reference is expanded. Thus, to rewrite the 'D' macro above so that it is evaluated to the literal string 'define(\$1,\$2)', the definition is:

```
DEFINE(D, [[define($1,$2)])
```

In order to redefine the macro 'DEFINE' (for example, so that the Pascal keyword 'DEFINE' can be used) the following definition can be used:

## MACRO

```
DEFINE([DEFINE] , [[DEFINE]])
```

Both arguments get one level of brackets stripped when the definition is processed; the second argument gets another level stripped when the macro is invoked.

The following built-in macros are provided:

**DEFINE(a,b)** defines a to be b and returns the null string.

**IFELSE(a,b,c,d)** returns c if a is identical to b. Otherwise, it returns d.

**INCR(a)** interprets a as an integer and returns a+1.

**SUBSTR(a,m,n)** returns a substring of the string a starting at character number m and extending for n characters.

**LEN(a)** returns the length of a.

**INCLUDE(a)** returns the contents of file a.

**EXPR(a)** returns the result of evaluating infix expression a. Operators in increasing order of precedence are as follows. Parentheses may be used as usual.

&	logical OR and AND
!	unary logical NOT
== ^= <= < > >=	arithmetic comparison
+ -	addition and subtraction
* / %	multiplication, division, modulus (remainder)
**	exponentiation
+ -	unary plus and negation

Logical operators return 0 (false) or 1 (true)

### *DIAGNOSTICS*

**arith evaluation stack overflow**  
Arithmetic expressions can only be nested to 30 deep.

**arg stack overflow**  
The total number of arguments exceeds the limit of 100.

**call stack overflow**  
Definitions can only be nested to 20 deep.

**EOF in string** An end-of-file has been encountered before a bracketed string has been terminated.

**evaluation stack overflow**  
Too many characters are used for the name, definition, and arguments of one macro. 2500 characters are allowed.

**unexpected EOF** An end-of-file was reached before a macro definition was terminated.

filename: can't open

The named file could not be opened.

filename: can't include

The indicated file cannot be included with the built-in macro INCLUD.

includes nested too deeply

Files included with the built-in macro INCLUD can be nested only up to 128 deep.

expression: invalid infix expression

There is a syntax error in the indicated infix expression as passed to the built-in macro EXPR.

too many characters pushed back

A macro expansion is too large to be rescanned. A macro definition may contain up to 2500 characters.

name: too many definitions

The table space for macro definitions has been exhausted; this occurred upon the definition of the named macro.

token too long

A name or symbol in the input was longer than the token buffer. Each token may be up to 200 characters long.

## MRGRGY (MERGE\_REGISTRIES)

**MRGRGY (MERGE\_REGISTRIES) -- Merge two network registries.**

### FORMAT

**MRGRGY rgy1\_path rgy2\_path**

MRGRGY merges two previously independent network registries. This command merges each network's site directories. Thus, all site directories' person, project, organization, full name, and account files are merged.

MRGRGY selects one master registry file to be the merged master. It then merges the master registry files from each network into the merged master. The merged master contains the name of the new master registry file and the names of the site directories that were in each of the original master registry files. The other master registry file (i.e., the file that is not the merged master) is backed up and then deleted.

The MRGRGY command completes in four phases:

Phase	Description
1	MRGRGY creates the new merged PPO and account files and saves them in temporary files. Some .bak files are changed, but no changes are made to the original PPO and account files. If Phase 1 fails because MRGRGY detects duplicate entries in the PPO and account files, fix the problem and rerun MRGRGY. If Phase 1 fails due to a network problem, rerun MRGRGY.
2	MRGRGY saves the merged PPO and account files in the site directories listed in the first master registry file. If Phase 2 fails due to a network problem, rerun MRGRGY.
3	MRGRGY creates the merged master registry file and saves it in the first master registry file. MRGRGY deletes the second master registry file. If Phase 3 fails due to a network problem, recreate the original master registry files from your backup copies. Then rerun MRGRGY.
4	MRGRGY performs a SALRGY to copy the merged PPO and account files to all site directories in the merged master registry. (Some sites already have these files; they will get new copies.) If Phase 4 fails due to a network problem, run SALRGY.

Before you use MRGRGY, use CMPPO and CMACCT to check for all name and account collisions that would prevent MRGRGY from running to completion.

MRGRGY is for use in a DOMAIN/BRIDGE internet. For more information on merging registries, see *Managing DOMAIN Internets*.

## ARGUMENTS

**rgy1\_path**  
(required)

Specify the master registry file of one of the registries to be merged; this file will become the new, merged registry's master file.

The master registry file may be specified by its pathname, (`//node/REGISTRY/RGY_MASTER`), by a node's copy of the master registry file (`//node/REGISTRY/REGISTRY`), or simply by a node entry directory (`//node`). If a node's `/REGISTRY/REGISTRY` file or a node entry directory is specified, the `//node/REGISTRY/REGISTRY` file is used to locate the registry's master file.

**rgy2\_path**  
(required)

Specify the master registry file of the other registry to be merged.

## EXAMPLES

The following command merges two registries. The registry masters are `//alpha/registry/rgy_master` and `//beta/registry/rgy_master`.

```
$ mrgrgy //alpha/registry/rgy_master //beta/registry/rgy_master
```

Phase 1:

```
Merging the person files
Merge completed
```

```
Merging the project files
Merge completed
```

```
Merging the org files
Merge completed
```

```
Merging the full names files
Merge completed
```

```
Merging the account files
Merge completed
```

## MRGRGY (MERGE\_REGISTRIES)

### Phase 2:

Merged person file saved in registry //alpha/registry/rgy\_master's sites

Merged project file saved in registry //alpha/registry/rgy\_master's sites

Merged org file saved in registry //alpha/registry/rgy\_master's sites

Merged full\_names file saved in registry //alpha/registry/rgy\_master's sites

Merged account file saved in registry //alpha/registry/rgy\_master's sites

### Phase 3:

Creating the new registry's master file in "//alpha/registry/rgy\_master"  
New master file completed

### Phase 4:

Merged person file saved in all new registry's sites

Merged project file saved in all new registry's sites

Merged org file saved in all new registry's sites

Merged full\_names file saved in all new registry's sites

Merged account file saved in all new registry's sites

\$

**MTVOL (MOUNT\_VOLUME) -- Mount a logical volume.**

## FORMAT

**MTVOL disk\_type[unit] [log\_vol\_number] [pathname] [options]**

A logical volume is a named storage area on a disk. MTVOL mounts a logical volume, making the files and directories it contains accessible. Up to eight volumes (both physical and logical) may be mounted on a node at any time. Of the eight, no more than five of those volumes may be logical.

Before a new physical volume can be mounted for the first time, you must initialize it. See the INVOL (INITIALIZE\_VOLUME) command description for details.

## ARGUMENTS

**disk\_type**  
(required)                      Specify the type of disk on which the volume being mounted resides. Valid disk types are: W (Winchester), S (Storage Module), or F (Floppy).

**unit**  
(optional)                      Specify disk unit number (0 or 1). If you use this argument, the unit number must follow the disk\_type ID immediately: no blanks in between. For example, "S1" denotes storage module unit 1.

Default if omitted: 0

**log\_vol\_number**  
(optional)                      Specify the disk volume number. This is the same number that you assigned when you formatted the disk using INVOL. The first logical volume is numbered 1, the second 2, and so forth.

Default if omitted: 1

**pathname**  
(optional)                      Specify the name of the volume entry directory. This is the logical volume's top-level directory. Specify this pathname only if the entry directory is not already cataloged in the naming tree. If the pathname you choose already exists, an error will result.

Logical volume entry directories may appear anywhere in the naming tree, with one exception: if a logical volume entry directory is also the node's entry (top-level) directory, it must appear just below the network root directory (//).

If you omit the pathname argument, MTVOL assumes that the entry directory already exists, and searches the naming tree for it. If it finds the entry directory, MTVOL mounts the volume and prints the full entry directory pathname.

## MTVOL (MOUNT\_VOLUME)

If MTVOL does not find the entry directory, it prints an error message, and does not mount the volume. The search may fail for any of the following reasons:

- The entry directory has never been cataloged.
- The entry directory was uncataloged when the volume was last dismounted.
- The entry directory pathname exists on another node, for which directory information is currently unavailable.

An unsuccessful search does not mean that you cannot mount the volume. It simply means that the volume entry directory pathname does not exist on your node. To mount the volume, issue the MTVOL command and supply an entry directory pathname.

Even if the MTVOL finds the entry directory pathname, the mount may fail if the volume is corrupt for some reason and needs salvaging. In this case, MTVOL asks for permission to mount the volume. You should usually respond "no" to this request, then run the volume salvaging routine SALVOL. Once the volume has been salvaged, you may try to mount it again. If you mount a corrupt volume without salvaging it first, damage to files in that volume may result.

Default if omitted: see above

### OPTIONS

- F** Force -- Mount the volume whether or not it needs salvaging, and do not ask for permission.
- NQ** No query -- Suppress query if a volume needs salvaging. Instead, mount the volume ONLY if it does not need salvaging.
- PR** Protect -- Mount the volume with write protection. Any attempts to write on the volume will fail.

### CAUTION:

Before removing a floppy disk volume mounted with MTVOL, you MUST use DMTVOL to dismount it. Failure to dismount the volume could result in lost or corrupt information.

### EXAMPLES

```
$ mtvol f /masterfloppy      Mount floppy and make a new entry directory.
$ dmtvol f                   Dismount it.
$ mtvol f                    Remount it using the new entry directory.
Volume mounted, entry directory is "/masterfloppy"
$
```

**MVF (MOVE\_FILE) -- Move a file.**

## FORMAT

**MVF source [destination] ... [options]**

MVF moves a file to a different location in the naming tree. Its effect is identical to

\$ CPF source destination	copy source to destination
\$ DLF source	delete the source

MVF always retains the source ACL on objects moved.

## ARGUMENTS

<b>source</b> (required)	Specify name of file to be moved. Wildcarding is permitted.
<b>destination</b> (optional)	Specify new file location. This pathname may be a derived name. If 'destination' is a directory, the command moves the source file into that directory. Otherwise it creates the new file using the name specified.
	Default if omitted: copy source to current working directory

Multiple source/target pairs and wildcarding are permitted.

## OPTIONS

Default options are indicated by "(D)."

<b>-C</b>	<b>(D)</b>	Create the target file. If the target file already exists, an error will result.
<b>-R</b>		Replace target file with source file. Use this option if the target file already exists. If the file does not exist, this option works like -C.
<b>-DU</b>		Delete when unlocked. This option is useful with -R. If the object to be replaced is locked when MVF is invoked, the replace operation will be performed when the object is unlocked.
<b>-F</b>		Force deletion of destination object if you have 'p' (protect) rights, even if you do not have 'd' (delete) rights.
<b>-LF</b>		List files moved.
<b>-LDL</b>		List files deleted by -R option.
<b>-CHN</b>		Change the name of an existing destination file if required. This option modifies the meaning of -C and -R. If -C is

## MVF (MOVE\_FILE)

specified, this option causes any existing object with the destination pathname to be renamed prior to the move. If -R is specified, the destination object is renamed if it is in use and cannot be deleted.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### EXAMPLES

```
$ mvf //anger/sam/letter //mary -r
```

Move the file "letter" from the directory "//anger/sam" to the directory "//mary" and replace the current file.

ND (NAMING\_DIRECTORY) +- Set or display naming directory.

## FORMAT

ND [pathname]

The ND command sets or displays the name of the naming directory. The naming directory is provided so that you may use a tilde (~) as a shorthand feature in pathname specifications. It is also important since the system checks its COM subdirectory (~COM) as a part of the default command search operation. The naming directory is set to the login home directory at login.

## ARGUMENTS

**pathname**  
(optional)

Specify directory name to be used as the naming directory. ND also accepts the command line parser arguments "-" and "\*". If you specify a hyphen (-), ND looks to standard input for the directory name. An asterisk (\*) followed by the name of a file directs ND to look inside that file for the new naming directory name.

Default if omitted: display the name of the current naming directory.

## EXAMPLES

```
$ nd /paul/links      Set naming directory to "/paul/links".
```

After execution of this command, you can use a tilde (~) in place of '/paul/links' at the beginning of any pathname. Thus "~sausage" would be the same as "/paul/links/sausage".

NETMAIN (NETWORK\_MAINTENANCE)

NETMAIN (NETWORK\_MAINTENANCE) -- Analyze network maintenance stats.

## FORMAT

NETMAIN [options]

NETMAIN is a highly interactive, menu-driven program that lets you control the NETMAIN\_SRVR network maintenance server and analyze the data that NETMAIN\_SRVR produces. NETMAIN provides detailed help from its menus. See *Administering Your DOMAIN System* for a complete description of NETMAIN's features, instructions about using its menus, and details about interpreting its output.

## OPTIONS

Default options are indicated by "(D)."

- W[HELP] (D) Make sure window is large enough to display command menus and interactive help.
- WC[MD] Set window size smaller for command menus only. If you later decide that you want to see the helps, grow the window manually with <GROW> or CTRL/G.
- NW Do not change window size.

## EXAMPLES

1. \$ netmain Run NETMAIN in a window large enough to display command menus and interactive help.
2. \$ netmain -wc Run NETMAIN in a window large enough (but no larger) to display the command menus.

NETMAIN\_CHKLOG (CHECK\_NETMAIN\_LOGS) -- Clean up bad log files.

## FORMAT

NETMAIN\_CHKLOG *pathname*... [*options*]

When the NETMAIN\_SRVR program halts catastrophically (for instance, during a node reset), it can leave the log file it was writing in a corrupt, unusable state. NETMAIN\_CHKLOG determines whether the log is corrupt and, optionally, deletes corrupt files.

If the *pathname* you specify points to some kind of file other than a NETMAIN log file, that file will almost always be ignored: it will almost never be deleted as a corrupt log. On very rare occasions, another kind of file may look so much like a corrupt log that it might be deleted accidentally if you use both `-D` and the standard command option `-NQ` (no query). Thus you should use `-D -NQ` with extreme care.

## ARGUMENTS

<b>pathname</b> (required)	Specify the files to be checked. Multiple names and wildcarding are permitted; separate names with blanks.
-------------------------------	--

## OPTIONS

Default options are indicated by "(D)."

<code>-D</code>		Delete corrupt log files.
<code>-ND</code>	(D)	Do NOT delete anything.
<code>-L</code>	(D)	Describe every file analyzed.
<code>-NL</code>		Describe only corrupt log files.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ netmain_chklog 'node_data/net_log/?*
```

NETMAIN\_NOTE

NETMAIN\_NOTE -- Place message in network error log.

## FORMAT

NETMAIN\_NOTE string [string ...]

NETMAIN\_NOTE sends a text string to NETMAIN\_SRVR, the network maintenance server. The message is broadcast to all maintenance servers.

Typical topics of maintenance notes include known or explainable network failures, scheduled down-time, and node installations.

## ARGUMENTS

**string**  
**(required)**

Specify message to be sent. You may use any string that is legal in a Shell command. (Note that the Shell takes special action on some keywords, such as 'if', unless you place them in quotes.) If there is more than one string, NETMAIN builds the note by concatenating the arguments that are separated by spaces.

## EXAMPLES

```
$ NETMAIN_NOTE 'Scheduled down time at 5 pm.'
```

```
$ NETMAIN_NOTE Cable disconnected at //sancho_panza
```

**NETSTAT (NETWORK\_STATISTICS) -- Display network statistics.**

## FORMAT

**NETSTAT [options]**

This command writes a summary of network and hard disk activity to standard output.

## OPTIONS

If no options are specified, NETSTAT returns a brief summary of network usage information for the current node.

- L** Long Report -- Provide more information than the summary.
- CONFIG** Configuration Report -- Display only node-specific hardware information: CPU type, display type, etc.
- N node\_spec ...** Provide information on specified node(s). See the section on node specifications in Chapter 3 for more information. Multiple 'node\_spec' strings are permitted; separate them with blanks.
- A** Report on all nodes in the network.
- R [n]** Repeat the NETSTAT command every n seconds until halted by CTRL/Q. Only counts that have changed at each iteration are displayed, and the values represent the amount of change rather than absolute values. The default value for 'n' is 10 seconds.
- S [n]** Send 'n' test messages to every node being listed (except the current node) before every repeat of the display. If this option is specified, -R must also be specified. This option provides a minimum amount of network activity during the wait time between NETSTAT repeats. The default value for 'n' is 100 messages.
- SAVE pathname** Save all statistics in the file named 'pathname'.
- SINCE pathname** Display counts that have changed since statistics were saved in 'pathname'.

## EXAMPLES

```
1. $ netstat
   The node ID of this node is 1FB.

   **** Node 1FB **** //diskless_$0001Fb diskless to //anger

   Up since 1983/02/01 at 8:17:06 Up for 1 day 2 hours 58 mins 4 secs
```

## NETSTAT (NETWORK\_STATISTICS)

```
Net I/O: total= 94625 rcvs = 66912 xmits = 27713
Winchester I/O: total= 0 reads= 0 writes= 0 {NOTE 1}
System configured with 1.0 mb of memory.
```

### 2. \$ netstat -L

The node ID of this node is 1FB.

```
**** Node 1FB **** //diskless_$0001Fb diskless to //anger
```

```
Up since 1983/02/01 at 8:17:06 Up for 1 day 2 hours 58 mins 52 secs
Net I/O: total= 94756 rcvs = 67010 xmits = 27746
```

```
10436 page-in requests issued.
6473 page-out requests issued.
41134 page-in requests serviced.
12139 page-out requests serviced.
```

```
Detected concurrency violations -- read: 0 write: 0
```

Xmit count	27746	Rcv eor	0
NACKs	272	Rcv crc	767
WACKs	1639	Rcv timeout	0
Token inserted	65	Rcv buserr	0
Xmit overrun	0	Rcv overrun	0

Xmit Ack par	3	Rcv xmit-err	3042
Xmit Bus error	0	Rcv Modem err	0
Xmit timeout	90	Rcv Pkt error	45
Xmit Modem err	0	Rcv hdr checksum	0
Xmit Pkt error	377	Rcv Ack par	10

Delay switched OUT.

```
Winchester I/O: total= 0 reads= 0 writes= 0 {NOTE 1}
```

Not ready	0	Contrlr busy	0
Seek error	0	Equip check	0
Drive time out	0	Overrun	0

CRC error percentage: 0.00%

```
Last ring hardware failure detected by node 241 {NOTE 2}
on 1983/02/02 at 10:05
System configured with 1.0 mb of memory.
A total of 0 ECC errors were detected.
```

### Notes on Examples

1. Node 1FB is running diskless, hence the absence of Winchester disk I/O activity.
2. At 10:05 A.M. on Feb. 2, 1983, the network cable was disturbed immediately upstream of node 241. This information, coupled with the network topology available from LCNODE (LIST\_CONNECTED\_NODES), can help you pinpoint a hardware malfunction.

NETSVC (NETWORK\_SERVICE) -- Set or display network services.

## FORMAT

NETSVC [options]

NETSVC sets or displays the network services that this node will perform. All changes take place immediately.

## OPTIONS

If no options are specified, NETSVC displays the network services allowed for this node.

Default options are indicated by "(D)."

- N                      None -- Disable all network services and physically disconnect this node from the network.
- L                      Local -- Allow only network requests originating at this node.
- R                      Remote -- Allow only network requests originating at other nodes.
- A                      (D)    All -- Allow both locally and remotely initiated network requests. (The size of the remote paging pool is not changed.)
- S[ERVERS] [n]        Servers -- Set the number of network servers to run on this node. At system start-up, the number of network servers is 1. If this node is a network partner for diskless nodes or has several remote file users, their performance can be improved by increasing the number of servers. If n is not specified, the maximum number of servers (3) is used.
- P [n]                 Pool -- Set local memory pool size. Network page requests originating at remote nodes may not use more than 'n' pages of the local node's memory. If n is not specified, all of the local node's memory is eligible for remote page requests.
- NET [net\_id]         Network ID -- Set or display network ID. Use this option to change or examine the ID of the network to which the node is attached. It affects only the node at which you type the command, not the rest of the network. Specifying a hexadecimal network ID changes your node's network ID. Using -NET with no argument forces NETSVC to display your network ID even if it is set to zero.

This option is useful only when there are no internet routers active on the node's network. Routers give the network ID to nonrouting nodes every 30 seconds, and may override the network ID you specify with this option.

## NETSVC (NETWORK\_SERVICE)

**CAUTION:** If the network ID you set with `-NET` differs from the network ID used by other nodes on your network, your node may not be able to communicate with those other nodes.

**CAUTION:** Be careful when revoking network access with `-N` or `-L`. Remote file users may have problems, and writable files may be damaged. If your node was the network partner for a diskless node, that node will crash when your node leaves the network.

Use the `-S` option carefully. Although you can increase the number of servers, you cannot decrease it. The only way to return to a smaller number of servers is to reboot the node. Also note that increasing the number of server processes decreases the number of user processes allowed. When you run `NETMAN`, the number of servers increases from one to two. If the number of servers is already two or greater when you start `NETMAN`, it will not increase further.

### EXAMPLES

```
$ netsvc
Network operations allowed: ALL
Number of network servers: 1
Remotely initiated paging pool limit: NONE
Network ID: 437A9
$
```

**NEXT** -- Return to the top of a loop.

## FORMAT

### NEXT

NEXT interrupts the flow of control in a Shell loop construct (FOR, SELECT, and WHILE). When NEXT is encountered in a FOR or WHILE loop, control passes back to the top of the loop (see examples below). When NEXT is encountered in a SELECT loop, control passes to the next CASE clause. (This is useful when you have specified SELECT ONEOF but want to test multiple things under certain circumstances).

You may terminate the flow of control in a loop by using the EXIT command. See the EXIT command description for more information.

The NEXT command requires no arguments or options.

## EXAMPLES

Consider the following section from a Shell script:

```
n := 0
while ((^n < 10))
do read -type integer n
   if ((^n < 10)) then next endif
   args ^n
enddo
```

As long as the READ command reads integers into variable "n" that are less than 10, the NEXT command executes and causes the script to return to the top of the WHILE loop. When the value of n is greater than or equal to 10, the script prints the number then leaves the WHILE loop and continues execution.

For more information on variables, refer to the *DOMAIN System User's Guide*.

NOT

NOT -- Negate a Boolean value.

## FORMAT

**NOT command**

NOT takes the Boolean value returned by a command or expression and negates it. This is useful primarily with the program control structures (IF, WHILE, etc.) used in Shell scripts.

## ARGUMENTS

**command**

(required)

Specify a command or expression that returns a Boolean value.

## EXAMPLES

Assume the following lines appear inside Shell scripts.

```
#
# Loop as long as no error file exists.
#
while not existf error_file
do args "No error file yet ..."
enddo
# End of script
```

```
#
# Verify user response.
#
eon
read -p "Type the pathname of the file to be deleted: " name
read -p "Are you sure you want to delete ^name?" verification
if ((^verification = "yes")) then
    delete := true
else
    delete := false
endif

if (( not ^delete )) then
    args "^name not deleted."
else
    dlf ^name -l
endif
# End of script
```

OBTY (OBJECT\_TYPE) -- Set or display the type of an object.

## FORMAT

OBTY *pathname...* [*object\_type*]

OBTY is intended for system-level debugging use only. Misuse of this command can cause objects to become inaccessible and programs to behave incorrectly.

## ARGUMENTS

**pathname**  
(required) Specify object whose type is to be set or displayed. Wildcarding of this pathname is permitted.

**object\_type**  
(optional) Specify new type setting. 'Object\_type' must be one of the following:

NIL	(nil type UID)
UASC	unstructured ASCII (text) file
REC	streams records file
HDRU	streams header-undefined file
OBJ	object file
PAD	display manager pad
SIO	sio descriptor file
UNDEF	undefined file type
NULLDEV	null device (bit bucket)
DDF	device descriptor file (for GPIO)
MBX	mailbox
AREA	D3M area file
SCH	D3M object (sub)schema file
MT	magnetic tape descriptor file
BOOT	system bootstrap file

Executable files (output of compilers and binders) are OBJ. Most other binary files are REC.

Default if omitted: display current type of 'pathname'

## OPTIONS

OBTY accepts no special options of its own, although it does use the command line parser, and so accepts the standard command options listed in Chapter 3.

## EXAMPLES

The sequence of the following commands is significant.

```
$ obty testfile          Display current object type.
"testfile" object type is nil.
$ obty testfile uasc     Set type to "uasc".
```

OBTY (OBJECT\_TYPE)

```
$ obty testfile  
"testfile" object type is uasc.
```

Display new object type.

**OS (OVERSTRIKE) -- Convert ASCII to FORTRAN carriage control.**

## FORMAT

**OS [pathname...]**

OS converts a file containing ASCII carriage control (for such things as form feeds and backspacing for underlining) into a file that can be printed on a line printer with FORTRAN carriage control. By default, output is written to standard output; redirect it into a file with the ">pathname" expression.

If you create a new file containing the overstruck text, OS automatically sets the file's carriage control flag so that printers we supply will interpret the file correctly. If you use OS in a pipeline, however, the flag is not set (since output goes to standard output). In this case, you must use the -FTN option on the PRF command for the file to be printed correctly. See examples 2 and 3 below.

## ARGUMENTS

**pathname**  
(optional)

Specify file to be converted. Multiple pathnames are permitted, separated by blanks. All output is concatenated, however.

Default if omitted: read standard input

## EXAMPLES

- |   |  |
|---|--|
| <pre>1. \$ os mary    \$</pre>  | <p>Convert the file "mary" and write to standard output.</p>   |
| <pre>2. \$ fmt letter   os &gt;letter.overstruck    \$ prf letter.os      -npag    \$</pre> | <p>Format the file "letter", pipe output to OS, and write the results into "letter.os." This file is then printed on the default printer.</p>            |
| <pre>3. \$ fmt letter   os   prf -npag -ftn    \$</pre>                                     | <p>Format the file "letter" and pipe it directly to the line printer. Note the use of "-ftn" to ensure that proper carriage control is used.</p>         |
| <pre>4. \$ fmt letter   prf -npag -pr spin    \$</pre>                                      | <p>Format "letter" and print it on a Spinwriter printer. Since Spinwriters use ASCII carriage control, OS and the -FTN option on PRF are not needed.</p> |

PAGF (PAGINATE\_FILE)

PAGF (PAGINATE\_FILE) -- Paginate a file.

## FORMAT

PAGF [options] [pathname...]

PAGF paginates the named files to standard output. Each file is printed as a sequence of pages. Each page is 66 lines long by default, including a 6-line header and 3-line footer. The header includes the file name, the date and time, and the page number.

## OPTIONS

**-L n** Set the page length to 'n' lines. The default page length is 66 lines.

## ARGUMENTS

**pathname**  
**(optional)** Specify file to be formatted. Multiple pathnames are permitted separated by blanks.

Default if omitted: read standard input

## EXAMPLES

```
$ pagf -L 20 mary >mary.short
```

Paginate the file "mary" into pages 20 lines long and write them to "mary.short".

**PPRI (PROCESS\_PRIORITY) -- Set or display process priority.**

## FORMAT

**PPRI [process\_name...|-UID uid] [options]**

The process priority is an integer ranging from 1 (low) to 16 (high). When the operating system decides which process to run next, it chooses the process that currently has the highest priority. As a process executes, its priority increases as it waits for events (such as keyboard input) and decreases as it computes for long periods without waiting. By default, the priority is bounded by the range 3 through 14 when a process is created. The PPRI command lets you change these bounds to any other numbers in the range of 1 to 16.

A forked process inherits the priority settings of its parent process.

## ARGUMENTS

**process\_name...**  
(optional)

Specify name of process whose priority is to be set or displayed. Multiple names and wildcarding are permitted. If the process does not have a name, use the -UID option (below).

Default if omitted: use current process.

## OPTIONS

If no options are specified, the current priority bounds are displayed.

- LO n** Set priority lower boundary. 'n' must be in the range 1-16 inclusive. If omitted, the lower boundary is set to 3.
- HI n** Set priority upper boundary. 'n' must be in the range 1-16 inclusive. If omitted, the upper boundary is set to 14.
- U[ID] uid** Specify the UID of an unnamed process whose priority is to be set or displayed.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

1. `$ ppri` Display defaults for current process  
`MY_SHELL: minimum_priority = 3, maximum priority = 14`
2. `$ ppri process_7 -lo 1 -hi 4` Restrict process\_7 to low priorities
3. `$ ppri -lo 12 -hi 12` Current process will always have priority 12

PRF (PRINT\_FILE)

PRF (PRINT\_FILE) -- Queue a file for printing.

## FORMAT

PRF [pathname...] [options]

PRF queues a file for printing. The file must be an ASCII stream (i.e., text) file, a graphics metafile (GMF), or a GPR bitmap object. After successfully queuing a file, PRF displays a message containing the full pathname of the file that you queued.

You can execute PRF once for each file that you want to print (specifying all the necessary options every time), or you can enter PRF's interactive mode and hand files to the program continuously. See the examples below.

Files queued by PRF are physically printed using PRSVR (PRINT\_SERVER).

When you invoke PRF, it first sets all options to their default states (as described below). Next, it looks for a PRF configuration file called `~USER_DATA/PRF.DB` unless you have invoked PRF with the `-NDB` option (described below). If PRF locates a configuration file, it executes the options contained in the file to configure the current session. Finally, it proceeds to process any options on the command line or in the interactive session.

A menu-based version of the PRF command is also available. See the PRFD command description for more information.

### *A Word About Printronix Carriage Control*

The Printronix line printer that we supply can interpret two kinds of carriage control: ASCII (the default for DOMAIN files) and FORTRAN. In files with ASCII carriage control, the NEWLINE and FORM FEED characters have special meanings. NEWLINE terminates a line. FORM FEED causes a page eject if it is the only character in a record. In any other position, FORM FEED causes unpredictable results.

If a file has FORTRAN carriage control (which consists of special characters in the first column of every line in the file), PRF works one of two ways. If the file has been produced by the OS (OVERSTRIKE) command or by one of your own programs that explicitly sets the file's FORTRAN carriage control flag, PRF automatically treats every record as a line, and interprets the first character in the record as a carriage control character with its standard FORTRAN meaning (i.e., '1' signals a form feed, '+' signals a carriage return without a line feed, etc.). Within a program, you can set the carriage control output flag by calling `STREAM_$REDEFINE`. To generate a file with FORTRAN carriage control from a file formatted with FMT, use the OS command.

If you know that the file has FORTRAN carriage control, but aren't sure whether or not the file's carriage control flag has been properly set, specify `-FTN ON` (below) to force use of this feature.

## ARGUMENTS

**pathname**  
(optional)

Specify the file to be printed. Multiple pathnames and pathname wildcarding are permitted.

Default if omitted: read standard input.

## OPTIONS

The following options may appear on the Shell command line or in PRF interactive mode as noted below. In addition, you may place one or more options in a configuration file (see -CONFIG). In that case, create the file with one option per line without the prefixed hyphens (-). See Example 3 below.

Many options described below have default values that are "specified in the PRSVR configuration file." This is a file established by your System Administrator for each printer in the network. If you omit one of these options, your file will be printed using the value found in that configuration file. For example, omission of the -BANNER option may cause your file to be printed with a banner page, or without it, depending on the default established for the printer in question by its configuration file. For complete details about the contents of the print server configuration file, including default settings, see *Administering Your DOMAIN System*.

If no options are specified, the file(s) are printed using ASCII carriage control, with pagination enabled, on the default printer (as established by PRSVR).

*The following options apply to all file types.*

**-INTER[ACTIVE]**

Enter interactive mode.

**-SEARCH\_DIR {ON|OFF}**

Search through all the directories of all the active processes on your node for the file(s) to be printed. This option is most useful in interactive mode, when the working directory of the PRF process may be different from the working directory of the file to be printed. This option is OFF by default.

**-COP[IES] n**

Print multiple copies of the file, where 'n' is the requested number of copies. If -COP[IES] is specified, 'n' is required. If this option is omitted, one copy is printed by default.

**-PR[INTER] name**

Specify the printer 'name' for printing the file. This option is useful only if more than one printer is in use on the network, or if a printer has been assigned a nonstandard name with the "PRINTER\_NAME" configuration directive in the PRSVR command. If you omit this option, PRF uses the default printer name, "P". Note that "P" is also the default printer name used by the PRINT\_SERVER.

**-S[ITE] entry\_dir**

Specify print queue (/SYS/PRINT) on alternate node by giving that node's entry directory name. This option allows you to

maintain more than one printer queue directory. You may want to maintain separate queues for different organizations, or you may want two queues to provide redundancy in case of node failure.

- C[OPY] (D)** Place a copy of the print file in the /SYS/PRINT/SPOOLER directory.
- NC[OPY]** Do not copy the print file into the /SYS/PRINT/SPOOLER directory. In this case, the print server uses the file itself rather than a duplicate. This option is only available with print servers running under SR9.5. If this option is specified, then -ND becomes default, and the file is NOT deleted unless -D is explicitly specified as well.
- D[ELETE] (D)** Specify that the print file is deleted when the print server is finished printing it.
- ND[ELETE]** Do not delete the print file when the print server is finished printing it. This becomes default if -NC is specified.
- USER[\_NAME] name**  
Specify user name that will appear on the banner page of the printed file. The alarm facility of PRF also uses this name to determine who should be notified when printing is complete (see -SIG below). This means that this name must be a valid login name (unless you don't care about sending an alarm). If this option is omitted, the current login name is used.
- SIG[NAL] {ALARM|OFF}**  
Request an alarm server signal when the file has finished printing. The default is OFF.
- BAN[NER] [ON|OFF]**  
Enable/disable banner page. The default is specified in the PRSVR configuration file. If neither ON nor OFF is specified, ON is assumed.
- CONFIG[\_FILE] [pathname]**  
Specify a file containing further PRF options, one per line. Do not use prefixed hyphens (-) with the option names in the configuration file. If 'pathname' is omitted, PRF will execute the configuration file ~USER\_DATA/PRF.DB.
- NDB** Suppress processing of the configuration file.
- TEXT** Specify text mode for printing ASCII files. This is the default print mode.
- PLOT** Specify plot mode. Include this option to print bitmap files created by a graphics metafile (GMF) manager or GPR, or the CPSCR (COPY\_SCREEN) command.
- TRANSPARENT** Specify that when the file is printed, the records of the file are to

be passed directly to the printer driver routine with no processing by the print server.

**-PAPER\_SIZE {A|B|LEGAL|A3|A4|A5|B4|B5}**

Select the paper size. You must specify one of the following size codes:

Code	Size in inches (mm)
----	-----
A	8.50 x 11.00
B	11.00 x 17.00
LEGAL	8.50 x 14.00
A3	11.69 x 16.54 (297mm x 420mm)
A4	8.27 x 11.69 (210mm x 297mm)
A5	5.38 x 8.27 (137mm x 210mm)
B4	9.84 x 13.90 (257mm x 364mm)
B5	5.93 x 9.89 (182mm x 257mm)

This option is only available for the DOMAIN/LASER26 and APPLE LASERWRITER printers. PRF assumes that the correct size of paper is in the printer's paper tray. Therefore, you must check the paper tray before printing. The default paper size is specified in the PRSVR configuration file.

*The following options apply to text files only.*

- NPAG**                    Disable the headers and margins generated by PRF.
- MARGINS [ON|OFF]**  
Enable/disable margins generated by PRF. If this option is specified without ON nor OFF, ON is assumed. The default is 'ON'.
- TOP n**                    Specify page top margin, in inches. The default is a value specified in the PRSVR configuration file.
- BOT[TOM] n**             Specify page bottom margin, in inches. The default is a value specified in the PRSVR configuration file.
- RIGHT n**                Specify page right margin, in inches. The default is 0 inches.
- LEFT n**                 Specify page left margin, in inches. The default is 0 inches.
- HEADERS [ON|OFF]**  
Enable/disable page headers and footers generated by PRF. If this option is specified without ON nor OFF, ON is assumed. The default is specified in the PRSVR configuration file.
- HEAD[\_STRING] l-string/c-string/r-string**  
Specify contents of left, center, and right components of the page header generated by PRF. Components may be empty strings. The following special characters return the values indicated when they appear in the header strings.

## PRF (PRINT\_FILE)

**e** = escape character  
**#** = current page number with 1 leading and 1 trailing space  
**%** = current date  
**!** = filename  
**&** = filename's last time, date modified  
**\*** = insert a space in text string (literal spaces are not allowed)

Example: `-HEAD !/Page#/%` will produce a header with the filename in the left component, the string "Page" followed by the current page number in the center component, and the current date in the right component. The default header is a string specified in the PRSVR configuration file.

### **-FOOT[\_STRING] l-string/c-string/r-string**

Specify contents of page footers. The format is the same as for `-HEAD` above. There is no default footer.

### **-FTN [ON|OFF]**

Enable/disable FORTRAN carriage control. `-FTN ON` causes the `PRINT_SERVER` to use FORTRAN forms control even if the file does not have the FORTRAN carriage control flag. Use of this option will cause PRF to interpret the first character of each line as a FORTRAN carriage control character (and not print it). This can be unfortunate if the file has ASCII carriage control, so be careful. `-FTN OFF` causes the `PRINT_SERVER` to print the contents of column one rather than trying to interpret it as FORTRAN forms control. If this option is specified without `ON` nor `OFF`, `ON` is assumed. The default is 'OFF'.

### **-WRAP [ON|OFF]**

Enable/disable automatic line wrapping. When enabled, PRF will wrap any lines that exceed the right margin onto the next line. When disabled, PRF truncates lines that exceed the right margin. If this option is specified without `ON` nor `OFF`, `ON` is assumed. The default is 'OFF'.

*The following options are for use with printers supporting variable font and pitch sizes.*

### **-PITCH n**

Set the pitch (characters/inch) at which you wish the document to be printed. The following pitch settings are available on the printers indicated.

Printronic	10
Spinwriter	12
Imagen	8.5, 10, 12, 15, 17.1
GE	10, 12, 13.1, 16.7
Versatec	12
LaserWriter	1 to 100
Laser26	1 to 100

### **-POINT n**

Set the point size for the font to be used. This is a real number in units of a point which is 1/72 inch.

**-WEIGHT value**

Set the weight of the font to be used. This option is only valid for the GE printer type. Possible values are 'light', 'medium', and 'bold'. The default is 'medium'.

**-LQ [ON|OFF]**

Specify that the document is to be printed in 'letter quality' (ON) or in 'draft' (OFF) mode. This option is only valid for the GE printer type. If this option is specified without ON nor OFF, ON is assumed. The default is 'OFF'.

*The following options apply to plot files.*

**-RES[OLUTION] n**

Specify resolution of output plot in dots per inch. If you specify a resolution not available on the particular printer, the file is printed at the closest available resolution. The default resolution is specified in the PRSVR configuration file.

**-WHITE[\_SPACE] n**

Specify amount of white space (in inches) to appear between multiple plots in one file. The default is three inches.

**-BW[\_REV] [ON|OFF]**

Enable/disable black and white reversal for bitmaps. If this option is specified without ON nor OFF, ON is assumed. The default is 'OFF'.

**-MAGN[IFICATION] n**

Specify bitmap magnification value. 'n' is an integer in the range -1 to 16.

-1                   selects auto-scaling to magnify the bitmap to fill the available page space.

0                    selects 'one-to-one' scaling between the display and the printer for GMF bitmaps. (For GPR bitmaps, this translates to magnification 1.)

1-16                selects magnification by that amount. Portions of the magnified bitmap that exceed the printer page boundaries are clipped.

The default is 0.

*The following options apply only for files sent to printers that contain the PostScript interpreter, such as the DOMAIN/LASER26 and APPLE LASERWRITER printers.*

**-POST[SCRIPT] [ON|OFF]**

Enable/disable PostScript interpretation. When enabled, the data is passed through the PostScript interpreter. When disabled, the data is printed as text, plot, or transparent data. If this option is specified without ON nor OFF, ON is assumed. The default is 'OFF'.

## PRF (PRINT\_FILE)

- COL[UMNS] {1|2}** Specify single or double column printing. The default state is single column.
- LPI n** Specify the line spacing factor. 'n' is an integer indicating the number of lines per inch. The default state is six lines per inch.
- ORIENT[ATION] {PORT[RAIT]|LAND[SCAPE]}** Select the page orientation. PORTRAIT specifies that the text or x-axis of the bitmap is printed parallel to the short edge of the paper. LANDSCAPE specifies that the text or x-axis of the bitmap is printed parallel to the long edge of the paper and perpendicular to the short leading edge. The default state is PORTRAIT.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## COMMANDS

Once PRF has been invoked in interactive mode (see **-INTER** above), it accepts the following interactive commands at the "PRF> " prompt (in addition to the options already discussed).

- P[RINT] pathname** Queue the specified file for printing.
- Q[UIT]** Quit interactive mode and return to the Shell.
- SH[ELL]** Create a shell command line. This command allows you to issue Shell commands without leaving PRF interactive mode. When you have finished entering Shell commands, type CTRL/Z. This will return you to PRF interactive mode. Your previous PRF option settings remain undisturbed by the intervening Shell commands.
- INIT[IALIZE]** Reset PRF parameters to their default values.
- R[EAD] [printer]** List entries in the queue for the specified printer at the current site (as specified by -S). If 'printer' is omitted, then the contents of the queue (determined by the current setting of -PR) are listed.
- WD [pathname]** Execute the Shell command WD (WORKING\_DIRECTORY) to set or display the working directory.
- GET option** Display the value of the PRF option specified. Use this command to show the settings of the various PRF parameters.
- CAN[CEL] [queued\_filename]** Cancel printing of the specified file at the current site (as specified by -S). Note that you must specify the pathname which PRF assigns when the file is queued (which may

differ from the name of your original file). Use the READ command to display the names of currently queued files. If the filename is omitted, the last file to be queued by this process is cancelled. This command is only effective for files which have not yet physically begun to print.

## EXAMPLES

```
1. $ prf mary -npag -ftn          Queue "mary": suppress
   "//NODE1/MY_DIR/MARY" queued for printing.  pagination; force
   $                                          FORTRAN carriage
```

```
2. $ prf filex -s //tape
   "//NODE1/MY_DIR/TEST_FILE.PAS" queued for printing at site //TAPE.
   $                                          Queue "filex" to the
                                          printer queue on the
                                          node called //tape.
```

3. Configuration File: the following commands might appear in the default PRF configuration file `~USER_DATA/PRF.DB`.

```
PR ge
SITE //rye
FOOT %/my_file/&
```

4. Sample interactive session:

```
$ PRF -INTER
PRF> get pr
pr = p
PRF> -pr cx
PRF> get pr
pr = cx
PRF> -pitch 20
PRF> print test_file.pas
   "//NODE1/MY_DIR/TEST_FILE.PAS" queued for printing.
PRF> q
$
```

5. Running PRF from an icon:

If you would like to run PRF interactively in a process devoted to it, you might place the following command in your `~USER_DATA/STARTUP_DM` file:

```
cp -i -c 'P' /com/prf -inter -n print_file
```

This will create a PRF process and turn its window into an icon using the print icon character in `(/SYS/DM/FONTS/ICONS)`. Issue the DM command `ICON` to change the icon window into its full-size format.

PRFD (PRF\_DISPLAY)

PRFD (PRF\_DISPLAY) -- Invoke menu-based PRF.

## FORMAT

### PRFD

PRFD invokes the menu-based version of the PRF (PRINT\_FILE) command. This interactive version runs a graphics interface inside the process window in which PRFD was invoked. You control the operation of PRFD by pointing the cursor at a desired menu item and pressing <M1> (if you have a mouse) or <F1> or the space bar if you do not have a mouse. Internal help is available by pointing at a menu item and pressing the <HELP> key (R6S).

PRFD requires no arguments or options on the Shell command line. To exit PRFD and return to the Shell, select the "QUIT" menu item.

PRFD performs exactly the same functions as the command-line based PRF. See the PRF command description for complete information on those various functions.

**PROBENET (PROBE\_NETWORK) -- Probe network and display error statistics.**

## FORMAT

**PROBENET [options]**

This command broadcasts packets to the diagnostic socket in all nodes, then requests error counts indicating the status the broadcast was received with. It compiles counts from every node in the topology list and reports them to standard output.

## OPTIONS

Default options are indicated by "(D)."

*Use one of the following three options to specify the list of nodes to display.*

**-A (D)** Probe all nodes responding to an LCNODE command. If the network is completely corrupted so that messages cannot make a complete pass, use one of the other two options to specify precisely which nodes to test.

**-T pathname** Probe the nodes listed in the topology file indicated. The file must contain one hexadecimal node ID per line. Any text following a space after the node ID is ignored. Comment lines may be inserted if they are prefixed with a '#' or '{'.

**-N node\_id ...** Probe the node(s) specified by the indicated hexadecimal node ID(s). A good choice of nodes to test is a set evenly spaced around the network.

*Use the following options to specify which test to run.*

**-S n (D)** Specify the total number of packets to be sent to each node. The default number of packets is 10. If 0 is specified for 'n', no test messages are sent, but statistics from each node will be collected.

**-R [n]** Repeat PROBENET cycle every 'n' seconds. If 'n' is omitted, the cycle is repeated every 10 seconds. When <RETURN> is typed at the input window, the send cycle is terminated immediately and the statistics are gathered and reported.

*Use the following options to specify which packets are sent.*

**-D data\_file** Specifies that the packets will be taken out of the specified data file instead of the standard built-in data pattern.

**-LEN n (D)** Specify the length (in bytes) of the data portion of the test packet, in bytes. The default length is 1024 bytes.

*Use the following options to control the level of detail in the statistics report.*

**-L** Print long (detailed) error counts if there were any errors (i.e., at least one XMIT ERRS or RCV ERRS).

## PROBENET (PROBE\_NETWORK)

- ERR** Print header for each test, but only statistics for nodes which returned errors (XMIT and/or RCV ERRS).
- MON fail\_lim** Print header for each pass, but only statistics on passes whose total failure count equal or exceed the 'fail\_lim' value.
- SENS threshold** Open a window pane and select some output lines to append to this pane. The nodes selected are the ones whose error count exceed a five node running average error count by the specified threshold value. Also, all nodes with modem errors are appended to this pane. The use of this secondary output is to do some data reduction and pick the nodes at or near points of data corruption in the network. The window pane is also stored in a named pad file: PROBENET.PANE

### REPORTED STATISTICS

The following statistics are printed for each node:

- ATTEMPT** Number of probenet packets received.
- ERRS** Number of probenet packets received with errors. An increase in this count over previous nodes narrows the network problem between this and the previously displayed node.
- MODEM ERRS** Number of transmit or receive modem errors encountered by the node.
- BIPH** Number of transmit or receive biphasic errors encountered by the node. An increase in this count over previous nodes narrows the network problem between this and the preceding node, independent of probenet display.
- ESB** Number of transmit or receive elastic store buffer errors encountered by the node.
- TOKENS** Number of tokens inserted by this node. This statistic does not localize any problem.

### EXAMPLES

```
1. $ probenet {Probe the entire network once. No errors detected.}
There are 4 nodes in the test.
Broadcasting 10 1024-byte packets . 85/02/20 21:16:52 # failures = 0
Last Biph hardware failure detected by node 676 on 85/02/20 at 19:15
```

NODE	NAME	ATTEMPT	ERRS	MODEM ERRS	BIPH	ESB	TOKENS=	0
584	*diskless	10	0	0	0	0	0	Self
21	OS	10	0	0	0	0	0	
4A	HUBRIS	10	0	0	0	0	0	
3536	*diskless	10	0	0	0	0	0	

PROBENET (PROBE\_NETWORK)

2. \$ probenet -t node\_list -s 14400 -r 3600 -d data\_e3

{ Probes network and displays nodes specified in file "node\_list".  
This node broadcasts 14400 packets in 3600 seconds, i.e. four  
packets per second. The packet data comes out of file "data\_e3". }

There are 5 nodes in the test.  
Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds.  
85/02/20 21:58:19 # failures = 100  
Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

NODE	NAME	ATTEMPT	ERRS	MODEM		BIPH	ESB	TOKENS=	3
				ERRS	ERRS				
1967	GTX	14386	0	0	0	0	0	1	Self
15F5	SWI	14386	0	0	0	0	0	0	
2255	BIRDIE	14384	0	0	0	0	0	1	
3FD	FLASH	14386	3	3	3	0	0	0	
2B69	STANG	14385	3	0	0	0	0	1	

Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds.  
85/02/20 21:58:41 # failures = 100  
Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

NODE	NAME	ATTEMPT	ERRS	MODEM		BIPH	ESB	TOKENS=	3
				ERRS	ERRS				
1967	GTX	14383	0	0	0	0	0	1	Self
15F5	SWI	14383	0	0	0	0	0	0	
2255	BIRDIE	14381	0	0	0	0	0	1	
3FD	FLASH	14383	4	4	4	0	0	0	
2B69	STANG	14382	4	0	0	0	0	1	

{ Above example shows a problem between node 3FD and its predecessor  
in the network. }

PRSVR (PRINT\_SERVER)

**PRSVR (PRINT\_SERVER) -- Start the Print Server.**

## FORMAT

**PRSVR [config\_file\_name] [options] [&]**

PRSVR executes the print server program, which prints files submitted to the print queue with PRF (PRINT\_FILE). *You only need to execute this command when you start the node connected to the printer. Do not execute the command at other nodes: this will cause print files to be lost.*

The print server must run independently of user login, or it will stop when you log out. To start the print server, include this command line

CPS /COM/PRSVR [pathname]

in the printer node's STARTUP file.

PRSVR first performs some internal initialization, then, after about ten seconds, prints an introductory message on the printer. It then scans the print queue (/SYS/PRINT/QUEUE), and if it finds a file intended for it, prints the file. After printing all the files in the queue, it checks the queue for more files every 10 seconds.

For complete details about user-supplied device drivers for auxiliary devices and the contents of the print server configuration file, see *Administering Your DOMAIN System*.

## ARGUMENTS

**pathname**  
**(optional)**

Specify printer configuration file.

Default if omitted: use file PRINTER\_CONFIG.DATA.

**&**  
**(optional)**

Create separate Shell process in which to run the print server. This process is created without the normal pads or windows, thus running invisibly in the background. When the print server is started in this fashion, it stops automatically at logout. To stop this background process before logout, use the SIGP (SIGNAL\_PROCESS) command.

## OPTIONS

**-N name**

Specify a name for the print server process. If this option is not specified, the process is named PRINT\_SERVER.printername.

**PST (PROCESS\_STATUS) -- List process internal state information.**

## FORMAT

**PST [options]**

PST lists internal state information for all processes in the system by name or UID.

## OPTIONS

- R[EPEAT] n** Repeat every n seconds. If you include this option, the first pass displays the total time elapsed since process creation. Subsequent passes display changes from the previous pass, as shown in the first example below.
- N[ODE] node\_spec** Specify remote node whose process statistics are to be listed. See the section on node specifications in Chapter 3 for more information.
- UN** Display DOMAIN/IX process IDs.
- PA[GING]** Display process paging information. The paging data presented is private page faults, global page faults, disk paging I/O, and network paging I/O.
- C** Display only brief information on level 2 (user) processes. This output also suppresses the header lines and the processor time total. See Example 5 below.
- TY[PE]** Shows whether each process is a USER process (stops at logout), a SERVER process (started via -CPS), or an AEGIS process (internal to the operating system).

## EXAMPLES

1. \$ pst -r 20

```
-----|
Processor |PRIORITY| Program|State|   Process
Time (sec)|mn/cu/mx| Counter|   |   Name
-----|
 273.345 16/16/16   18088 Wait display_manager
 441.379  1/15/16 <active> Ready process_1
  68.086  1/16/16   17E90 Wait digital_clock
   7.531  1/14/16   17E90 Wait alarmer
   0.715  1/15/16   17C30 Wait mbx_helper
   0.715  1/15/16   17F78 Wait mail
   0.715  1/16/16   17F78 Wait shell
-----|
792.489
```

PST (PROCESS\_STATUS)

```

-----|
Processor |PRIORITY| Program|State|   Process
Time (sec)|mn/cu/mx| Counter|   |   Name
-----|
    0.262 16/16/16   18088 Ready display_manager
    0.262 1/16/16 <active> Ready process_1
    0.262 1/15/16   17E90 Wait digital_clock
    0.000 1/14/16   17E90 Wait alarmer
    0.000 1/15/16   17C30 Wait mbx_helper
    0.000 1/15/16   17F78 Wait mail
    0.000 1/16/16   17F78 Wait shell
-----|
    0.786

```

2. \$ pst -n //eve Display stats of node "//eve".

```

-----|
Processor |PRIORITY| Program|State|   Process
Time (sec)|mn/cu/mx| Counter|   |   Name
-----|
1016.261 16/16/16   18088 Wait display_manager
    7.269 1/ 2/16   17C30 Wait netman
    4.385 1/15/16   17C30 Wait mbx_helper
10.939 1/16/16   17F78 Wait server_process_manager
-----|
1038.855

```

3. \$ pst -un

```

-----|
Processor |PRIORITY| Program|State|UNIX INFORMATION |   Process
Time (sec)|mn/cu/mx| Counter|   | PID | PPID| PGID|   Name
-----|
1016.261 16/16/16   18088 Wait    1    0    1 display_manager
    7.269 1/ 2/16   17C30 Wait   23    1   23 netman
    4.385 1/15/16   17C30 Wait   24    1   24 mbx_helper
10.939 1/15/16   17F78 Wait   26    1   26 server_process_manager
-----|
1038.855

```

4. \$ pst -pa

```

-----|
Processor |PRIORITY| Program|State|Private| Global|D I S K| N E T| Process
Time (sec)|mn/cu/mx| Counter|   | Faults| Faults|Page IO|PageIO| Name
-----|
288.549 16/16/16   18088 Wait    2168  783    0  4180 display_manager
444.001 1/15/16 <active> Ready  6284 1557    0 11097 process_1
    7.793 1/15/16   17E90 Wait    279  120    0   436 alarmer
    0.715 1/15/16   17C30 Wait     36   6    0    84 mbx_helper
    1.502 1/16/16   17F78 Wait    104  51    0   175 mail
    0.715 1/16/16   17F78 Wait     44  22    0    74 shell
-----|
    815.296                               9039  2565  0  16197

```

5. \$ pst -c

```

1433.801 16/16/16   1F6A2 Wait    display_manager
    20.293 3/13/14   1F192 Wait    netman
    4.658 3/13/14   1F192 Wait    mbx_helper
    69.609 3/ 6/14   475BE8 Ready   base_shell
    37.487 3/10/14   1F352 Wait    alarm_server

```

6. \$ pst -ty

Node: BAD1

Time: Monday, August 26, 1985 2:23:06 pm (EDT)

Processor Time (sec)	PRIORITY mn/cu/mx	Program Counter	State	Type	Process Name
563587.995	-- -- --	-----	-----	aegis	<Null Process>
4189.242	-- -- --	-----	-----	aegis	<Aegis Processes>
15597.507	16/16/16	1F6A2	Wait	user	display_manager
958.367	3/ 3/14	1F602	Wait	user	process_31
4.633	3/14/14	1F192	Wait	server	mbx_helper
215.347	3/14/14	1F602	Wait	user	alarm_server
658.725	3/13/14	1F44A	Wait	user	clock
933.019	3/ 9/14	<active>	Ready	user	process_34
370.728	3/ 3/14	1F602	Wait	user	process_36
586515.828					

RBAK (READ\_BACKUP)

RBAK (READ\_BACKUP) -- Restore or index a magnetic media backup file.

## FORMAT

RBAK {pathname [-AS disk\_pathname] ... | -ALL} [options]

RBAK restores a magnetic media backup file which was written with WBAK (WRITE\_BACKUP). Use WBAK and RBAK to backup disks and to transfer information between separate DOMAIN installations. (Use the RWMT (READ\_WRITE\_MAGTAPE) command to transfer information to and from non-DOMAIN installations.)

RBAK operates in either "index" or "interchange" mode. To restore objects to disk, use interchange mode (-INT). To list object names on standard output, without restoring any information to disk, use index mode (-INDEX).

When using RBAK, please note the following:

- RBAK must be run on the node which is connected to the tape or floppy drive unit. You may accomplish this either by physically typing the RBAK command on the host node, or by running RBAK in a process on the host node created from your own remote node using the CRP CREATE\_PROCESS command.
- There is no special tape mounting command. Simply mount the tape on the tape drive and execute RBAK.
- Only one tape unit can be connected to any node.
- The directories on disk must have delete access to be replaced by directories from the backup media. The disk directories must have append access for files to be added from the backup media.
- Files on disk must have delete access to be replaced by files from the backup media.
- Locked objects cannot be restored from backup media. See the LKOB (LOCK\_OBJECT) command.

## ARGUMENTS

**pathname**  
(optional)

Specify name of object to be indexed or restored to disk. This may be a directory, file, or link. If the object is being restored, the new disk object will have the same name. If you wish the disk file to be saved under a different name, use -AS (below). Multiple pathnames are permitted; however, wildcarding is not supported.

Default if omitted: must use -ALL.

## OPTIONS

Default options are indicated by "(D)."

### *Backup File Identifiers*

One of the following options is required.

- F file\_no**            Read the backup file with the file number specified. You assigned this number with WBAK.
- F CUR**                Begin reading at current position on the backup media.
- FID file\_id**         Read the backup file name specified. You assigned this name using WBAK.

### *Mode Control*

- INT**            (D)    Select "interchange" mode. Backup files are restored to disk.
- INDEX**         Select "index" mode. Backup file names are listed on standard output; no information is restored to disk.

### *Catalog Control*

- ALL**                Restore or index all the objects in the backup file specified. This option is required if you do not use the 'pathname' argument to indicate a particular object to be indexed or restored.
- AS pathname1**      Restore the object specified and assign a different disk pathname ('pathname1'). This option is only valid when used with the 'pathname' argument on the RBAK command line.
- CR**                (D)    Specify create mode. RBAK does not restore objects if their names already exist on disk. It prints an error message if a name exists on both disk and backup media, and continues.
- R**                 Specify replace mode. RBAK deletes the existing disk object, and replaces it with the object read from backup media.
- FORCE**            Force object deletion if you have owner rights, even if you don't have delete rights.
- DU**                Delete when unlocked. If the object to be deleted is locked when RBAK is invoked, the delete operation will be performed when the object is unlocked.
- MS**                Specify merge-source mode. Similar to replace mode. If an object already exists on disk, RBAK deletes the disk version and restores the backup media version (the source). However, if the object is a directory, RBAK merges the backup media directory's contents with the disk directory.

## RBAK (READ\_BACKUP)

### **-PR pathname...**

Preserve specified objects on the disk. Multiple pathnames and wildcarding are permitted. If the objects exist on disk, they will NOT be overwritten by backup media versions. This option must be used with the -MS option.

### **-MD**

Specify merge-destination mode. Similar to create mode. If an object already exists on disk (the destination) RBAK does not restore the backup media version, and retains the disk version. However, if the object is a directory RBAK merges the backup media directory's contents with the disk directory.

### *Label Control*

**-SLA (D)** Display the backup media file label on standard output.

**-NSLA** Do not display the backup media file label.

### *Listing Control*

You may include the -L option, or any combination of -LD, -LF and -LL.

**-L** Write all the file, directory, and link names to standard output.

**-LD** Write all directory names to standard output.

**-LF** Write all filenames to standard output.

**-LL** Write all linknames to standard output.

### *Backup Device Control*

**-ANYS** Force RBAK to accept any section of the backup file. When a backup file spans multiple backup media volumes, RBAK normally begins with the backup media volume containing the backup file's first section, and proceeds to the backup media volume containing the second section, and so on. If you know which backup media volume contains the object you want to restore or index, use this option. This lets RBAK start at any section of the backup file.

**-REO** Force previous volume to be reopened, and suppress reading of backup media volume label. Use only when backup media has not been repositioned since last WBAK or RBAK.

**-DEV d[unit]** Specify device type and unit number. 'd' must be either 'M' (for reel-to-reel magnetic tape), 'CT' (for cartridge tape), or 'F' (for floppy), depending on which drive is being used. 'unit' is an integer (0-3). Both are required for reel-to-reel tapes (i.e., -DEV M2). A unit number is NOT required for floppy disks and cartridge tapes (i.e., -DEV F). If this option is omitted, RBAK assumes device M0.

**CAUTION:** Floppy disk support for this command is limited. In particular, error detection during reads and

writes is poor. DO NOT use this command with floppy disks when the data being placed on the floppy disks are critical and unrecoverable.

- RETEN                    Retension the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.
- NRETEN    (D)        Do not retension the cartridge tape.
- REWIND                Rewind the cartridge tape after reading or indexing. If this option is omitted, the cartridge tape is left positioned to the next tape file. This option is valid ONLY for the cartridge tape; reel-to-reel tapes get rewound automatically when removed from the drive.

#### *ACL Control*

- DAACL        (D)        Assign the destination directory's default ACL to the object being restored.
- SACL                Retain the restored object's original ACL.

#### *DTM/DTU Control*

- PDT                Preserve the object's original date-time modified and date-time used.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

### EXAMPLES

- \$ rbak -f 1 fred/soup        Read "fred/soup" in backup file 1 and restore it to disk. "Fred/soup" may be a directory, file, or link.
- \$ rbak -f 1 fred/soup -as //node5/noodle        Restore "fred/soup" and place it in "noodle" on node5.
- \$ rbak -dev ct -rewind        Rewind the cartridge tape prior to removing it from the tape unit.

### DIAGNOSTICS

#### *I/O Errors*

When RBAK encounters an I/O error, it attempts the operation again, for a total of five times. After the fifth attempt fails, RBAK prints out an error message describing which type of error occurred. If the error happened during an attempt to read from a tape, RBAK skips the tape block which it could not read, and tries to read the next one. (Note that a tape block is 8 Kb

## RBAK (READ \_BACKUP)

long.) If it again fails, after five tries, it skips that block and tries the next. This process will continue for a total of twenty consecutive failed blocks, at which time RBAK aborts.

tape rewind error

An I/O error occurred.

tape write-filemark error

An I/O error occurred.

tape space-filemark error

An I/O error occurred.

tape space-record error

An I/O error occurred.

i/o recovery failed

An I/O error occurred and the tape drive could not reposition for another try.

tape i/o error

An I/O error occurred.

tape i/o error -- data lost

An unrecovered I/O error occurred and data was lost.

### *Operator Errors*

first label on volume is not VOL1 label

Expected a standard label, and did not find one. The tape was not written with WBAK.

label version number in VOL1 label is not "3"

The label format is incorrect. The tape was not written with WBAK.

wrong volume, file header is inconsistent with previous trailer

The wrong continuation tape was put on the drive. This error can occur only when a multi-tape file is used.

magtape drive is offline

You have not put the drive on line.

tape is write-protected

The write enable ring is not on the tape.

file not found

The tape file specified was not found.

invalid unit number

Tape unit specified is not connected. Presently, only DEV M0 is supported.

pbu is not present.

No tape unit is connected to the node. RBAK can only be run on the node connected to the tape drive.

**RDYM (READY\_MESSAGE) -- Set system ready message.**

## FORMAT

**RDYM {-ON | -OFF}**

RDYM enables or inhibits the output of a system ready message to standard output after execution of each Shell command. The message lists the CPU time required to execute the command and the elapsed time since the last command. Both times are reported in seconds and decimal fractions of seconds. The message appears on a line following the echoed command line in the Shell's process transcript pad.

Turning on the ready message interactively or in a Shell script causes it to be printed after each command of the program is executed. If the ready message is not disabled at the end of the Shell script, it will remain in effect after the Shell script exits.

Ready message printing is enabled and disabled for *levels*. The level number increases each time a Shell script or the Shell is invoked, and decreases when it exits. The times printed in the ready message reflect the CPU and real time used since the last message at the same level. Thus, for example, if the ready message is enabled in a Shell script, after the last command of the program is finished, two ready messages will be printed: one showing the time used by that command, and one showing the time used by the whole Shell script.

If the ready message is turned on by one level, it will remain on when that level exits; however, if it is disabled by a level, it will revert to its previous state when that level exits.

By default, system ready messages are disabled at login.

## OPTIONS

<b>-ON</b>	Enable message.
<b>-OFF</b>	Disable message.

## EXAMPLES

```
$ rdyd -on
cpu time: 750.685.  real time: 5914.532.
$ bldt
AEGIS, revision 6.0, built on Friday, April 15, 1983 9:26:30 am (EST).
cpu time: 0.234.  real time: 1.736.
$ rdyd -off
$
```

## READ

**READ -- Set variables equal to input values.**

## FORMAT

**READ [options] {-TYPE type var\_name ... | variable\_list}**

The READ command reads input values and sets a list of variables to those values. The values from the input line are parsed as separate tokens (they must be separated by spaces), and each variable in the list is assigned the value of a token.

Use the "-P prompt" argument to instruct READ to issue a prompt. If you do not input values for all the variables names listed as part of the READ command, READ displays a "<more>" prompt to request further input.

By default, the type of each variable specified in the READ command depends on the type of each input value. You can, however, use the -TYPE argument to specify the individual type(s) of the variables.

## ARGUMENTS

**variable\_list**  
(optional)

Specify the names of the variables that receive the input values.

Default if omitted: must specify -TYPE (below).

## OPTIONS

**-TYPE type var\_name...**

Specify the type of the input value(s) that can be assigned to the particular variable name(s). Multiple variable names are permitted, separated by blanks. Once you specify a type in a particular READ command, READ assigns that type to all subsequent variable names, until you change the type specification. Valid types are

STR[ING]	character strings
INT[EGER]	integer numbers
BOOL[EAN]	Boolean values
ENV[IRONMENT]	environment variables
ANY	any type (the default)

If the type of the input value does not match the type specified for that variable name, READ issues an error and asks you enter another input value. Use -TYPE ANY to restore the Shell to its default state. In this case, it determines the proper variable type automatically.

Specifying -TYPE ENV var\_name causes the variable to become an environment variable. Environment variables are of primary concern to DOMAIN/IX users; please consult the DOMAIN/IX documentation for details about their usage.

**-P[ROMPT] prompt**

Specify a particular prompt string to request the input values. Enclose the string in single quotes if it contains literal blanks.

**-ERR[IN]**

Read input from error input instead of standard input. This option is useful for reading user input from the Shell's input pad (where error input is normally directed) when the READ command appears inside a pipeline, since standard input in that case is connected to the pipe.

**EXAMPLES**

Consider the following command line in a Shell script:

```
READ -P "Enter model and class:" model class
```

In this example, READ displays the prompt "Enter model and class:" in the process input window, and assigns the input values to the variables named "model" and "class", in that order.

The following section illustrates how the -TYPE option works. (The numbers in parentheses are used to refer to the different parts of the example.)

```
$ READ -P '> ' -TYPE integer tens ones -TYPE string number (1)
> 40 four
<non-integer 'four'; please reenter> > 4 (2)
<more> > forty-four (3)
$
$ LVAR (4)
integer tens = 40
integer ones = 4
string number = forty-four
$
```

In line (1) we define the prompt to be "> ", specify variables "tens" and "ones" of type "integer", and specify variable "number" of type "string". This means that the READ command expects its input to be three variables of types integer, integer, and string, in that order. When we enter the non-integer value "four", READ cannot assign this value to variable "ones", and issues the error message and prompt shown in line (2). In line (3) READ prompts for the third input value. The LVAR command, issued in line (4), displays the type, name, and value of the variables.

Here is a final example.

```
$ date | chpat ', ' '' | (read day month date year; readln time)
$ lvar
string time = 12:40:42 pm (EST)
integer year = 1985
string month = January
string day = Wednesday
integer date = 2
$
```

In this example, the output from the DATE command is piped to CHPAT, which removes the commas and then sends its output to READ and READLN where the proper variable assignments are made.

## READC

**READC -- Set variables equal to input characters.**

### FORMAT

**READC [options] variable\_list**

The READC command reads single characters as input, and sets a list of variables equal to those character values. READC parses each character from the input line as a separate token, and each variable in the list is assigned the value of a token. Use the "-P <prompt>" argument to instruct READC to issue a prompt.

The READC command considers all input to be type "string".

### ARGUMENTS

**variable\_list**  
(required) Specify the names of the variables that receive the input values.

### OPTIONS

**-P[ROMPT] prompt**  
Specify a particular prompt string to request the input values. Enclose the string in single quotes if it contains literal blanks.

**-ERR[IN]**  
Read input from error input instead of standard input. This option is useful for reading user input from the Shell's input pad (where error input is normally directed) when the READC command appears inside a pipeline, since standard input in that case is connected to the pipe.

### EXAMPLES

Consider the following sequence of commands and input:

```
$ readc -p "Do you want to continue? (y/n): " ans
Do you want to continue? (y/n): y
$ lvar
string ans = y
```

In this example, READC displays the prompt "Do you want to continue? (y/n): " in the process input window, and assigns the value of the first input character ("y" in this case) to the variable named "ans".

For more information on Shell variables, refer to the *DOMAIN System User's Guide*.

**READLN -- Set a variable equal to an input value**

## FORMAT

**READLN [options] variable\_list**

The READLN command reads a line of input and sets a variable to that value. Use the "-P <prompt>" argument to instruct READLN to issue a prompt. READLN accepts multiple variable names.

The variable type is always a string.

Refer to the descriptions of the READ and READC commands for related information.

## ARGUMENTS

**variable\_list**  
(required)

Specify the name(s) of the variable(s) that receives the input value(s). If you specify more than one variable name (separated by blanks), READLN assigns the values of input lines to the variables in the order that the variables were named.

## OPTIONS

**-P[ROMPT] prompt**

Specify a particular prompt string to request the input value. Enclose the string in single quotes if it contains literal blanks.

**-ERR[IN]**

Read input from error input instead of standard input. This option is useful for reading user input from the Shell's input pad (where error input is normally directed) when the READLN command appears inside a pipeline, since standard input in that case is connected to the pipe.

## EXAMPLES

Consider the following command line in a Shell script:

```
READLN -p "Enter total here: " total
```

In this example, READLN displays the prompt "Enter total here: " in the process input window, and assigns the value of the input line to the variable named "total."

RETURN

RETURN -- Return from current Shell level.

## FORMAT

RETURN [options]

The RETURN command causes the Shell to return from its current level with the specified status severity. See the ABTSEV command description for details about status severity levels.

## OPTIONS

Specify one of the following options to select the return severity level. All options must be specified in UPPERCASE letters.

Default options are indicated by "(D)."

-OK		Set level to OK.
-T[TRUE]	(D)	Set level to true.
-F[FALSE]		Set level to false.
-W[ARNING]		Set level to warning.
-E[RROR]		Set level to error.
-O[UTINV]		Set level to output invalid.
-I[NTFATAL]		Set level to internal fatal error.
-P[GMFLT]		Set level to program fatal error.
-M[AX_SEVERITY]		Set level to maximum severity error.

## EXAMPLES

The following lines are a portion of a Shell script:

```
#
# Test to see if the second parameter passed to the script is valid.
# If it is not, abort the script.
#
if eqs ^1 '-test' then
  if eqs ^2 '-b' then
    cpf ^3 temp.mss
  else
    args "(?) >>> ^2 <<< is not a valid parameter."
    return -P
  endif
else
  cpf ^1 temp.mss
```

endif

REVL (REVERSE\_LINES)

REVL (REVERSE\_LINES) -- Reverse each line in a file.

## FORMAT

REVL [pathname ...]

REVL copies the named files to standard output, reversing the order of the characters in every line.

## ARGUMENTS

pathname  
(optional)

Specify name of file containing lines to be reversed.

Default if omitted: read standard input.

## EXAMPLES

1. 

```
$ revl
```

This command produces interesting results.

```
.stluser gnitseretni secudorp dnammoc sihT  
*** EOF ***  
$
```

Reverse a line from standard input.
2. 

```
$ revl /sys/dict | srf | revl >rhyming_dict
```

```
$
```

Sort the system dictionary by suffixes to produce a rhyming dictionary.

**RTCHK (ROUTING\_CHECK) -- Test traffic between adjacent routers.**

## FORMAT

**RTCHK [options]**

RTCHK performs a simple test to verify that the router is able to pass packets to and from an adjacent router. RTCHK is for use in a DOMAIN/BRIDGE internet. You must be logged onto a routing node in order to use RTCHK.

Use the -DEVICE option to specify a network controller to test. You must give a device type (e.g. RING, IIC) to the device option. The RTSVC program, with no command line options, shows you which network devices your node has.

Older versions of RTCHK used a different command line syntax to specify the type of network hardware checked. The old command line options still work in RTCHK version 9.5, but are no longer supported.

For more information on RTCHK, see *Managing DOMAIN Internets*.

## OPTIONS

Default options are indicated by "(D)."

**-N net.node\_id**

Test packet transmission to and from the specified node. The network ID 'net' must be a network that the router touches. Not valid if -DEVICE is specified.

**-DEV[ICE] dev-name [dev-num]**

Test packet transmission over a specific network device. Use the RTSVC program to display the names (used for 'dev-name') and controller numbers (used for 'dev-num') of the network devices attached to your node. Not valid if -N is specified.

**-S n**

Specify the number of test packets to exchange with the other router. If -S is not specified, 10 packets are exchanged.

**-DAT**

**(D)**

Specify that each test packet carries 1024 bytes of test data.

**-NODAT**

Omit test data from the test packets.

## EXAMPLES

```
$ rtchk -n 3CE02A8.4851 -s 1000
```

Exchange 1000 test packets with node 4851 on network 3CE02A8. The router must be attached to that network.

```
$ rtchk -nodat
```

Exchange 10 short test packets with the other node attached to the IIC or T1 connection.

## RTCHK (ROUTING\_CHECK)

```
$ rtchk -dev iic -s 100
```

Exchange 100 test packets with  
the other node on the IIC or  
T1 network.

**RTSTAT (ROUTING\_STATISTICS) -- Display internet router information.**

## FORMAT

**RTSTAT [options]**

RTSTAT shows the behavior of an internet router at each of its network ports. RTSTAT is most useful in a DOMAIN/BRIDGE internet. It can, however, provide information about non-routing nodes as well as routing nodes.

For more information on RTSTAT, see *Managing DOMAIN Internets*.

## OPTIONS

- DEV** Report device-specific statistics for each port.
- NET [net\_id ...]** Report counts of references to each network specified. The reference counts for each network are roughly proportional to the number of packets transmitted towards the network, but may be somewhat higher. -NET with no arguments uses the list of visible networks.
- R [n]** Repeat every 'n' seconds. If 'n' is omitted, repeat every 10 seconds.
- N node\_spec ...** Report statistics for each node in the list.
- See the section on node specifications in Chapter 3 for more information. If this option is omitted, RTSTAT reports statistics for the local node only.
- DESC[RIBE]** Print a description, several lines long, of each statistic. The description appears only once for each statistic, the first time it is printed with a non-zero value.

## EXAMPLES

1. \$ rtstat

```
-----
1232.3D9      pkts routed:  110024   queue oflo:      0
               misrouted:      0     rt too far:     14

               RING      pkts sent:    73278     pkts rcvd:      72434

               IIC      pkts sent:    67830     pkts rcvd:      61077
```

RTSTAT (ROUTING\_STATISTICS)

2. \$ rtstat -net

---

1232.3D9	pkts routed:	110024	queue oflo:	0
	misrouted:	0	rt too far:	14
RING	pkts sent:	73278	pkts rcvd:	72434
	towards net:	1232	ref cnt:	74540
IIC	pkts sent:	67830	pkts rcvd:	61077
	towards net:	1234	ref cnt:	53532
	towards net:	1231	ref cnt:	9193
	towards net:	1233	ref cnt:	5105

**RTSVC (ROUTING\_SERVICE) -- Set or display internet routing service.**

## FORMAT

**RTSVC [-DEVICE dev-name [dev-number] [options]]**

RTSVC displays or alters the characteristics of a network port. RTSVC is for use in a DOMAIN/BRIDGE internet. You must be logged onto the node you wish to control in order to use RTSVC.

For complete information on RTSVC, see *Managing DOMAIN Internets*.

## OPTIONS

If no options are specified, RTSVC displays the characteristics of every active network. If you specify any other options, you must specify the type of network controller using a -DEVICE command line option.

You may use only one -DEVICE option on any command line:

**-DEV[ICE] dev-name [dev-number]**

Specify the network device type: RING, IIC, or USER (for EtherBridge routers). The device number applies only to USER devices. You may use the name ETHERBRIDGE in place of USER if you prefer. The dev-number option applies only to USER networks, and is required. Find the device number by using RTSVC without command line options (as shown in the examples).

Earlier versions of the RTSVC command used a different command line syntax for specifying network devices. The old command lines still work, but you should start using the new -DEVICE command lines as soon as possible. Future versions of RTSVC will not accept the older command lines.

This option changes the network ID of any network port:

**-NET net\_id** Assign the port a hexadecimal network ID number.

Only one of the following options may be specified at a time.

**-ROUTE** Allow routing service to or from the port.

**-NOROUTE** Allow normal AEGIS requests but no routing service.

**-OFF** No AEGIS requests or routing service allowed.

**-USER nn** Set an EtherBridge network. The value is not changed until the routing node is rebooted or the routing process is stopped and restarted.

## RTSVC (ROUTING\_SERVICE)

### EXAMPLES

```
$ rtsvc -device iic -net 007302ED -route
```

Assign a network ID to the Interphase controller and allow internet routing at that port.

```
$ rtsvc -dev ring -noroute
```

Stop internet routing through the ring port, but allow normal AEGIS requests for paging, file service, etc. Do not change the node's network ID.

```
$ rtsvc
```

Display the networks attached to this node.

Controller	Net ID	Service offered
=====	=====	=====
RING	76AO	Own traffic only
USER           46	768C	Port not open

The node in the last example touches two network: a Domain ring and an Ethernet, via the Etherbridge product. You need the device number information ("46") from this display in order to turn on routing at the EtherBridge network. Use the device number as shown here:

```
$ rtsvc -dev user 46 -route
```

"46" is the device number.

**RWMT (READ\_WRITE\_MAGTAPE) -- Read/write foreign magtapes.**

## FORMAT

**RWMT mode\_control [pathname...] [options]**

RWMT reads tapes from non-DOMAIN installations and writes tapes which can be read by non-DOMAIN installations. RWMT can read and write unlabeled tapes, as well as ANSI level 1-4 labeled tapes.

For information on reading and writing tapes intended for exchange with other DOMAIN installations, see the RBAK (READ\_BACKUP) and WBAK (WRITE\_BACKUP) commands.

When using RWMT, please be aware of the following:

- RWMT must be run on the node which is connected to the tape unit. You may accomplish this either by physically typing the RWMT command on the host node, or by running RWMT in a process on the host node created from your own remote node using the CRP (CREATE\_PROCESS) command.
- Only one tape unit can be connected to any node.

## ARGUMENTS

**pathname**  
(optional)

Specify name of file to be read from or written to tape. This argument is only valid with the -R and -W mode control options (below). Multiple pathnames are permitted. Wildcarding is permitted for write (-W) operations only.

Default if omitted: read pathnames from standard input.

## OPTIONS

Default options are indicated by "(D)."

### *Mode control*

One of the following mode control options must be specified. If you omit it, RWMT will prompt you for it. You may have RWMT prompt for all necessary options by using the -P option.

**-L[ABEL]**

Write ANSI X3.27-1978 volume label on a tape. This option causes RWMT to write an ANSI volume label and dummy file on the magtape volume. An optional owner and volume ID, which are stored in the volume label, may be specified (see -VID and -OWN below). This is the way to initialize a labeled tape; if any information existed on the tape, it is erased by this labeling operation.

## RWMT (READ\_WRITE\_MAGTAPE)

If you are labeling a tape, then the following two options may also be used.

**-VID vol\_id** Specify a 1-6 character volume ID for use when labeling a volume. This option is only valid when used with the **-L mode\_control** option (above). The default volume ID is '' (blank).

**-OWN owner\_id** Specify a 1-14 character owner ID for use when labeling a volume. This option is only valid when used with the **-L mode\_control** option (above). The default owner ID is '' (blank).

**-I[NDEX]** List objects on an ANSI-labeled physical tape volume. **-INDEX** produces a listing of all files or file sections on an ANSI-labeled physical tape volume. The contents of the physical volume (VOL1) label and all file header labels are written to standard output.

**-W[RITE]** Specify one or more disk files ('pathname' argument) to be written to tape. The default format is ANSI labeled, ASCII, fixed-length records of 80 bytes each, and 80-byte blocks. If desired, any of these parameters can be changed using the options described below. If more than one pathname is specified, the disk files are written to sequential tape files. Tapes written by RWMT are always in accordance with ANSI level 4 format.

Before writing a labeled file, the tape volume itself must be labeled with the **-LABEL** mode control option (above).

**-R[EAD]** Specify one or more tape files to be read from tape and stored on disk. **READ** reads one or more tape files and writes them to disk using the specified pathnames ('pathname' argument). The default tape file format is the same as that for the **WRITE** option. If the tape is labeled under ANSI level 2, 3, or 4, the file format (block length, record length, and record format) is read from the tape. If the tape is unlabeled, or labeled with ANSI level 1, you must specify the tape format using the options below. If more than one pathname is specified, adjacent tape files are read and stored under the specified pathnames.

### *Label Control*

**-ANSI** (D) Specify that the tape is labeled in conformance to ANSI X3.27-1978, level 1, 2, 3, or 4.

**-UNLAB** Specify that the tape is unlabeled.

### *Tape Format*

**-ASC** (D) Specify that all tape file contents are in ASCII characters.

- EBC** Specify that all tape file contents (except labels) are in EBCDIC characters.
- RAW** Specify that all tape file data is to be treated in raw form (see example 5).
- NPAR (D)** Specify no disturbance of parity bits when reading or writing data.
- PAR** Specify that parity bits should be forced off when reading data from tape and forced on when writing data to tape.
- RL reclen** Specify the maximum length, in bytes, of a record in the tape file. This option is only valid when used with either the -R or -W mode control options (above). It is unnecessary when reading an ANSI level 2, 3, or 4 file. The default record length is 80 bytes.
- BL blocklen** Specify the length, in bytes, of a physical tape block. This option is only valid when used with either the -R or -W mode control options (above). It is unnecessary when reading an ANSI level 2, 3, or 4 file. The default block length is 80 bytes.
- BF blockfac** Specify a blocking factor -- the number of records to store into or read from a physical tape block. This is an alternative to the -BL option, since the record length multiplied by the blocking factor yields the block length. This option is only valid when used with either the -R or -W mode control options (above). Using this option is only meaningful if your tape has fixed-length records. This option is unnecessary when reading an ANSI level 2, 3, or 4 file. The default blocking factor is 1.
- RF format** Specify record format. Valid values for 'format' are "F" (fixed-length records and blocks); "D" (variable-length records (this is ANSI 'D' format)); "S" (spanned records); or "U" (undefined record format). The default format is "F." Note that if you are writing a cartridge tape, only 512 byte blocks may be written; D, S, and U formats are not supported.

#### *Tape File Identifiers*

- FID file\_id** Specify a 1-17 character file ID to be written in the file header label for use when writing a file to a labeled volume. This option is only valid when used with the -W mode control option (above). If this option is omitted, the name of the file being written is used.
- F [position]** Specify the file position for -R or -W operations. Valid values for 'position' are "CUR," "END," or a nonzero integer value. A position of "CUR" selects the current tape position; the tape must have been previously read or written by RWMT and its position must not have been disturbed. This option is only valid when used with either the -R or -W mode control options (above).

## RWMT (READ\_WRITE\_MAGTAPE)

A position of "END" selects the end of the tape file set. This option is valid only when used with the -W mode control option, and causes RWMT to append the specified disk file ('pathname' argument) to the very end of the tape file set.

A position specified by a nonzero integer value selects the file at that absolute position in the tape volume. This option is only valid when used with either the -R or -W mode control options (above). If multiple 'pathname' arguments are supplied, the value of 'position' is incremented by one after each file has been read or written.

The default value for 'position' is 1.

### *Backup Device Control*

- DEV d[unit]** Specify device type and unit number. 'd' must be either 'M' (for reel-to-reel magnetic tape), 'CT' (for cartridge tape), or 'F' (for floppy), depending on which drive is being used. 'unit' is an integer (0-3). Both are required for reel-to-reel tapes (i.e., -DEV M2). A unit number is NOT required for floppy disks and cartridge tapes (i.e., -DEV F). If this option is omitted, RBAK assumes device M0.
- RETEN** Retension the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.
- NRETEN (D)** Do not retension the cartridge tape.

### *Miscellaneous Control Options*

- SBIN** Cause all stream files written to contain the binary attribute (normally, output stream files contain the ASCII attribute).
- P** Cause RWMT to prompt for all unspecified parameters.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

1. `$ rwmt -label -own "R and D" -vid "demo"` Initialize a tape with the given owner and volume ID.
2. `$ rwmt -w c?*_example -f 1 -rf d -rl 200 -bl 2048` Copy the  
32 records of "cmf\_example" written to tape file 1. wildcarded  
8 records of "cmt\_example" written to tape file 2. files to  
4 records of "cpboot\_example" written to tape file 3. tape.  
25 records of "cpf\_example" written to tape file 4.

3. `$ rwmt -index` List the files on the tape.

Volume label:

Volume ID: "DEMO " Owner ID: "R AND D " Access: " "

File/Section	File ID	Cr Date	Acc	RF	RL	BL
1 1	CMF_EXAMPLE	83/02/17		D	200	2048
2 1	CMT_EXAMPLE	83/02/17		D	200	2048
3 1	CPBOOT_EXAMPLE	83/02/17		D	200	2048
4 1	CPF_EXAMPLE	83/02/17		D	200	2048
5 1	CPT_EXAMPLE	83/02/17		D	200	2048

End of file set.

\$

4. `$ rwmt -r cpboot_example.tape -f 3` Copy tape file 3 to a  
 4 records read from tape file 3 into disk file named  
 "cpboot\_example.tape". "cpboot\_example.tape".

\$

5. RAW mode

RWMT permits a tape file to be read in RAW mode. In this mode, each block read from the tape is written into one record in a stream file, unmodified by the program. To read a file in RAW mode, you should specify the maximum record size using the `-RL` argument. If you do not, the default value of 80 bytes is used, and any records longer than that are truncated. Also, undefined record format should be used.

For example:

```
$ rwmt -r -f 1 -rf u -raw -rl 512 rawfile
```

reads tape file number 1 into "rawfile", with a maximum record length of 512 bytes.

Files may be written in the same manner:

```
$ rwmt -w -f 1 -rf u -raw -rl 512 rawfile
```

## DIAGNOSTICS

### *I/O Errors*

When RWMT encounters an I/O error, it attempts the operation again, for a total of five times. After the fourth retry fails, RWMT prints out an error message describing which type of error occurred. If the error was during an attempt to read from a tape, RWMT skips the tape block which it could not read, and tries to read the next one. If it again fails, after five tries, it skips that block and tries the next. This process may continue for a total of 20 consecutive failed blocks, at which time RWMT aborts. If the error occurred while writing a block to tape, RWMT aborts after the fifth attempt to write to that block.

#### tape rewind error

An I/O error occurred. An illegal operation may have been attempted, such as trying to perform an INDEX on an unlabeled volume.

#### i/o recovery failed

An I/O error occurred and RWMT could not position the tape for another try due to a separate I/O error.

## RWMT (READ \_ WRITE \_ MAGTAPE)

tape write-filemark error

An I/O error occurred.

tape space-filemark error

An I/O error occurred.

tape space-record error

An I/O error occurred.

tape i/o error

An I/O error occurred.

### *Format Errors*

The following errors indicate that the labels of a labeled tape do not conform to ANSI standards.

first label on volume is not VOL1 label

Expected a standard label, and did not find one.

label version number in VOL1 label is not "3"

The label format is incorrect.

a HDR1 label is missing where one is required

A file on the tape does not begin with the correct format.

an EOF1 (or EOV1) label is missing where one is required

A file on the tape does not end with the correct format.

a double filemark was encountered unexpectedly

The tape format is incorrect.

inconsistent file sequence numbers

The tape format is incorrect.

file sequence number tracking error

The tape format is incorrect.

variable record with invalid record control word encountered

The tape format is incorrect.

spanned record with invalid segment control word encountered

The tape format is incorrect.

erroneous spanning indicator or segment out of sequence encountered

The tape format is incorrect.

### *Operator Errors*

The following errors indicate that the equipment was not set up correctly or, that the tape was not loaded correctly.

wrong volume, file header is inconsistent with previous trailer

The wrong continuation tape was put on the drive. This error can occur only when a multitape volume is used.

magtape drive is offline

You have not put the drive on line.

tape is write-protected

The write enable ring is not on the tape.

*Miscellaneous Errors*

file not found

The tape file specified is not in the volume.

invalid record format specifier

The record format specified must be D, F, U, or S.

block size is too large

Block size specified is larger than 16 Kb.

conflicting blocking information

Record size specified is larger than block size specified for "F" record format.

invalid unit number

Tape unit specified is not connected. Unit numbers must be between zero and three inclusive.

pbu is not present

No tape unit is connected to the node. RWMT can only be run on the node connected to the tape drive.

This file contains n invalid records which were not written to tape

N records of length less than 4 were in the original disk file. Such records cannot be written to tape. This message is just a warning; the rest of the file was written to the tape.

SALACL (SALVAGE\_ACLS)

SALACL (SALVAGE\_ACLS) -- Salvage an Access Control List.

## FORMAT

SALACL [options] [volume]

SALACL salvages the ACL structure on the volume you specify. It corrects the reference counts on all ACL objects. Reference counts record the number of files that use each ACL. After it repairs reference counts, SALACL merges duplicate ACLs into a single copy, and deletes unused ACLs. You should run SALACL once a week or so unless you always receive reports that no reference counts needed repairing (i.e., everything is perfect).

SALACL cannot merge duplicate ACLs on files that are currently in use (locked) (for example, library files). This is not especially serious, as the duplication consumes very little disk space. To merge all possible ACLs on a node, including things like libraries, bring the node up diskless, mount the volume using MTVOL, and then run SALACL on the mounted volume.

## ARGUMENTS

**volume**

(optional)

Specify the entry directory pathname for the volume whose ACLs you intend to salvage. Note that SALACL cannot salvage a volume mounted on a remote node.

Default if omitted: '/' (node entry directory)

## OPTIONS

**-N[O\_]S[UMMARY]**

Suppress summary information.

**-V[ERIFY]**

Verify only; do not delete or merge any ACLs.

**-N[O-]M[ERGE]**

Do not merge duplicate ACLs into a single copy.

## EXAMPLES

```
$ salacl                               Salvage the ACLs of the current volume.
Warning: unable to merge two equivalent ACLs:
//grover/sys/node_data/Shell - object is in use (OS/file server)
ACL objects found:                      24
ACL reference counts fixed:             2
ACLs garbage collected:                 0
ACLs in use during g.c.:                0
ACLs merged with equivalents:           12
```

**SALD (SALVAGE\_DIRECTORY) -- Salvage a directory.**

## FORMAT

**SALD pathname...**

SALD corrects problems in directories caused by a system crash or network failure. The specified pathname must refer to a directory.

SALD makes the specified directory usable and saves as much information as possible.

The following are symptoms of damaged directories:

- LD reports that the directory is empty, but it cannot be deleted.
- LD lists a file in the directory, but no other command can find the file.
- The directory is completely unreadable, and errors occur on every access attempt.

## ARGUMENTS

**pathname**  
(required)

Specify name of directory to be salvaged. Multiple pathnames and wildcarding are permitted.

## OPTIONS

SALD has no unique options.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ sald /sqh/data_dir           Salvage the directory specified.
```

SALRGY (SALVAGE\_REGISTRY)

SALRGY (SALVAGE\_REGISTRY) -- Salvage (update) the network registry.

## FORMAT

SALRGY [options]

SALRGY maps the registry data files from all the sites in the network registry, determines the latest one and copies that version to all the sites. You will probably not be able to use this command unless you are the network administrator for your network, since SALRGY requires you to have write access to the network registry database. For complete information on the use of the network registry, see *Administering Your DOMAIN System*.

## OPTIONS

**-R pathname** Specify the node entry directory name of a node containing a registry file copy. If you omit this option, SALRGY obtains the registry pathnames from the current node's registry file copy (/REGISTRY/REGISTRY).

## EXAMPLES

```
$ salrgy -r //os      Salvage the network registry using the
$                      registry file copy on node "os".
```

SALVOL (SALVAGE\_VOLUME) -- Verify and correct allocation of disk blocks.

**FORMAT**

**From AEGIS: SALVOL**

**From Mnemonic Debugger: EX SALVOL**

Each logical volume is divided into disk blocks. SALVOL verifies and, if necessary, corrects the tables that describe the allocation of disk blocks to the files stored on the disk. SALVOL also returns to the free space pool all blocks that are no longer in use: those allocated to temporary files or to deleted portions of permanent files.

SALVOL can usually restore a disk after a system crash or an improper dismount of a volume. Please refer to the SALVOL description in the *DOMAIN System Utilities* manual for complete information on the use of this software tool.

SALVOL prompts for all required arguments and options.

SCRTO (SCREEN\_TIMEOUT)

SCRTO (SCREEN\_TIMEOUT) -- Set/show screen timeout

## FORMAT

SCRTO [n] [-NONE]

The SCRTO command sets or displays the number of minutes after the last input event that the Display Manager waits before it shuts off the display screen.

By default, the DM waits 15 minutes before it shuts off the display.

## ARGUMENTS

**n**  
(optional)                      Set the number of minutes for the DM to wait before it shuts off the display.  
Default if omitted: display current timeout setting.

## OPTIONS

**-NONE**                      Disable automatic timeout; never turn off the display.

## EXAMPLES

```
$ scrto                              Show initial setting.
The screen timeout is set to 15 minutes
$ scrto 10                            Set delay to 10 min.
$ scrto
The screen timeout is set to 10 minutes
$ scrto -none                        Disable timeout.
$ scrto
Screen timeout is currently turned off
$ scrto 15                            Restore initial setting.
$
```



SCRCH (SHOW\_SCREEN\_CHARACTERISTICS)

SCRCH (SHOW\_SCREEN\_CHARACTERISTICS) -- Show screen characteristics.

## FORMAT

SCRCH [fieldname...] [options]

SCRCH lists values of screen characteristics on standard output.

## ARGUMENTS

**fieldname**  
(optional)

Specify fieldname(s) of the graphic attribute to be described. If omitted, the default is to show values of all currently supported attributes.

The possible fieldnames are **x\_visible\_size** (visible x dimension screen area in pixels), **y\_visible\_size** (visible y dimension screen area in pixels), **n\_planes** (number of planes available), and **n primaries** (number of primaries for graphics, either 1 for monochrome or 3 for RGB color).

## OPTIONS

Default options are indicated by "(D)."

- V** Verbose description of field values.
- X** (D) Specifically give value of x visible screen size.
- Y** (D) Specifically give value of y visible screen size.
- P** (D) Specifically give value of the number of planes.
- PR** (D) Specifically give the number of primaries. The possible values are 1 for monochrome and 3 for RGB color.

## EXAMPLES

1. 

```
$ scrch
1024 800 4 3
$
```
2. 

```
$ scrch -v
x_visible_size - 1024
y_visible_size - 800
n_planes - 4
n primaries - 3
$
```
3. 

```
$ scrch -x -y
1024 800
$
```

## SCRCH (SHOW\_SCREEN\_CHARACTERISTICS)

4. \$ scrch -y -x  
800 1024  
\$
5. \$ scrch -p -x -v  
n\_planes - 4  
x\_visible\_size - 1024  
\$
6. \$ scrch x\_visible\_size  
1024  
\$



**SELECT -- Execute a SELECT statement.**

## FORMAT

```

SELECT arg [mode]
  CASE arg [TO arg]
    [command...]
  [CASE...
    command...]
  [OTHERWISE
    command...]
ENDSELECT

```

SELECT allows you to build a control structure that executes commands according to the results of one or more Boolean tests. The Shell uses each CASE clause to perform a separate Boolean test on the initial SELECT argument. If the CASE argument is equal to the SELECT argument, the result of the test is TRUE and the command(s) within the CASE clause execute.

You may test multiple CASEs simultaneously. If you place several CASE clauses on the same line (or specify line continuation with @<RETURN>), the CASEs are logically OR'd and return TRUE if any one of the CASEs is TRUE. (See example below.)

The (optional) TO clause allows you to specify an integer or string range for testing. For example, you might specify "CASE 0 TO 9" to test for any single digit, or "CASE a TO z" to test for a lowercase letter.

The (optional) OTHERWISE clause executes if and only if none of the CASE clauses returns TRUE (regardless of whether the selection mode was 'ONEOF' or 'ALLOF').

## ARGUMENTS

### arg

(required)

Any valid token, defined integer or string variable, or expression. SELECT compares the first 'arg' with each of the CASE 'arg's to determine which command(s) to execute.

### mode

(optional)

Specify selection mode. Valid modes are 'ONEOF' and 'ALLOF'. If you specify ONEOF (the default), SELECT executes only the first CASE statement that returns a TRUE value. If you specify 'ALLOF', SELECT executes all CASE statements that return TRUE.

Default if omitted: use 'ONEOF'.

### command...

(optional)

Specify the command to be executed when the CASE test returns TRUE. This may be a Shell command, a Shell script, a variable assignment, or any other valid Shell operation. Multiple commands are permitted; separate them with semicolons or NEWLINE characters.

## SELECT

Default if omitted: no command executed for this CASE clause.

### EXAMPLES

```
select ^a allop
  case 1 case 2 case 3
    args "This will print if ^a = 1 or ^a = 2 or ^a = 3"
  case 1 @
  case 2 @
  case 3
    args "This is the same test as the previous one, since the"
    args "carriage returns are escaped."
  case 4 # This is a case without a body to execute.
  case 5 to 10
    args "This will print if ^a is in the range 5-10."
  case 6
    args "This will also print if ^a = 6, since ALLOP is"
    args "specified."
  otherwise
    args "This will print if ^a is not between 1 and 10."
endselect
```

**SEND\_ALARM -- Send messages to alarm servers.**

## FORMAT

**SEND\_ALARM string ... {[target\_option ... ]} [-L]**

SEND\_ALARM transmits one-line messages that the ALARM\_SERVER can display. You can direct messages to:

- a particular user, specified by the name used for log-in,
- the user on a particular node, specified by the node's entry directory or node ID,
- the users on all of the diskless nodes that have a particular paging partner,
- or combinations of these categories of users.

You must always specify at least one user to receive the message, but you may use any of these techniques for choosing users.

Messages are not stored and message delivery is not guaranteed. If the intended recipient is not running an alarm server when the message is sent, or if other problems arise, the message is never delivered. SEND\_ALARM notifies you when messages can not be delivered.

## ARGUMENTS

**string ...**

**(required)**

Specify the message you want to send. Multiple strings are permitted, separated by blanks. If several strings are present, they are concatenated with intervening spaces to form the message.

## OPTIONS

**-L**

List the target nodes or users as the messages are sent.

At least one of the following target options must be specified.

**-U[SER] login\_name ...**

Specify one or more users to whom the message should be sent. If you use this option more than once, then several lists of users are concatenated. Every user on the concatenated list receives the message.

If a user is logged on several nodes at the same time, and has alarm servers running at each of those log-in sessions, only one of the alarm servers will be able to receive the messages.

**-N[ODE] node\_spec ...**

Transmit the message to whatever user is logged in on the

## SEND\_ALARM

specified node, if any. See the section on node specifications in Chapter 3 for more information. Multiple node specs are permitted; separate them with blanks. If you use this option more than once, then several lists of nodes are concatenated. Every user on a node on the concatenated list receives the message.

If several users are logged in on one node simultaneously and are running alarm servers, only one user is able to receive messages directed to that node.

### **-DI[SKLESS] node\_spec ...**

Transmit the message to whatever user is logged in on any diskless node whose host node is specified by 'node\_spec', as in the **-NODE** option. Multiple node specs are permitted. If you use this option more than once, then several lists of nodes are concatenated. Note that this option does not send a message to the user on the paging partner. See **-DIN** below.

### **-DIN node\_spec ...**

This option allows you to send a message to a disked node and all of its diskless partners. This is the same as specifying **"-NODE node\_spec ... -DISKLESS node\_spec ..."**.

### **-ME or -MYU[SER]**

Adds you to the list of users to receive the messages.

### **-MYN[ODE]**

Adds the node at which the command is entered to the list of target nodes.

### **-MYDI[SKLESS]**

Same as **"-DISKLESS node\_spec"**, where 'node\_spec' is the node at which the **SEND\_ALARM** command was entered.

### **-MYDIN**

Same as **"-MYNODE -MYDISKLESS"**.

### **-ALLN[ODE]**

Send the message to every node seen by an **LCNODE**.

## EXAMPLES

```
$ SEND_ALARM 'Meeting is cancelled' -USER joe_k sue_b john_f mary_g
```

```
$ SEND_ALARM Please log out. Node AAD going down. -DIN Oaad
```

```
$ SEND_ALARM Compilation completed -ME
```

```
$ SEND_ALARM Power going off at 17:30 tonight -ALLNODE -L
```

```
$ SEND_ALARM 'Loop 13 is being switched out' *loop13
```

**Note:** In this example, "loop13" is a file in the working directory containing further input for the **SEND\_ALARM** command line. It might, for example, contain the text:

SEND\_ALARM

-N //dirtball //thorazine  
-N //top 317F  
-U network\_manager  
-MYDIN

## SET

**SET -- Set current Shell conditions.**

### FORMAT

**SET [options]**

SET allows you to change the state of the current Shell. It accepts the same options as the SH (SHELL) command, thus permitting you to alter current Shell conditions without having to generate an entirely new Shell. See the SH command description for general information about Shells.

Separate Shell commands already exist for altering a few of the current Shell conditions (EON | EOFF; VON | VOFF; etc.). SET combines all of those functions into a single facility.

### OPTIONS

Default options are indicated by "(D)."

- B[ON]**                      Send the output of a background process (created with the & parsing operator) to the display. The output of the background process is displayed in the transcript pad of the Shell where it was invoked. If you do not specify -B, the output of the background process is sent to /DEV/NULL.
  
- BOFF**                      (D)      Do not display output from a background process.
  
- NB[ON]**                    (D)      Same as -BOFF.
  
- C[OMMAND] arg1 ...**  
Execute the following argument(s) as a Shell command, exactly as if it had been read as an input line. If any argument contains explicit blanks, enclose it in quotes. The Shell passes all text following -C to 'arg1' as arguments, so if you want to specify other options to the SH command itself, they must precede -C.
  
- E[ON]**                      Enable evaluation of variables outside of expressions. If -E is specified, the Shell always evaluates variables, regardless of the context in which they appear. If -E is not specified, variables are evaluated only inside variable expression delimiters, ((expression)); otherwise, the Shell treats the ^var\_name expressions as strings and they are not evaluated.
  
- EOFF**                      (D)      Evaluate variables only inside expressions.
  
- NE[ON]**                    (D)      Same as -EOFF.
  
- I[NTER]**                   Behave as though input is being entered interactively: prompt for each input line, and do not exit on errors or quit faults (DQ or CTRL/Q from keyboard). Normally, the Shell only executes interactively if its input comes from a pad or sio-line. Use of this option forces prompting.

- S[CRIP]T** (D) Behave as though executing a Shell script: do not prompt and abort on error. A Shell normally will not quit; any error or quit command is assumed to apply only to the last command given to the Shell.
- NI[NTER]** (D) Same as -S.
- N[EXECUTE]**  
Interpret each command line only; suppress execution.
- EX[ECUTE]** (D) Interpret each command line and execute it.
- P[ROMPT]1** *prompt\_string*  
Define the prompt string for the Shell created with SH.
- P[ROMPT]2** *subprompt\_string*  
Define the subprompt string for the Shell created with SH. (The subprompt appears when you continue a Shell command over more than one line).
- START** [*file*] (D) Execute the specified script after the Shell is created. If 'pathname' is not specified, the Shell searches the login home directory for a file called USER\_DATA/SH/STARTUP and executes it if it exists. No error occurs if that file does not exist.
- NSTART** Disable startup file execution.
- V[ON]** Display each line of text in the transcript pad as it is read by the Shell program.
- VOFF** (D) Disable input verification.
- NV[ON]** (D) Same as -VOFF.
- X[ON]** Display each command in the transcript pad immediately before execution. Each command is given in full, with its complete pathname and with the values of arguments inserted.
- XOFF** (D) Disable input examination.
- NX[ON]** (D) Same as -XOFF.

#### EXAMPLES

1. `$ set -pi 'Input> '` Change the current Shell's primary prompt to "Input> ". Note the use of quotes to preserve the trailing blank.
2. `$ set -eon -xon -von` Enable variable evaluation, command examination, and verification.

SETVAR (SET\_VARIABLE) -- Set the value of a variable

SETVAR (SET\_VARIABLE) -- Set the value of a variable

## FORMAT

SETVAR [options] {[**-TYPE** type] var\_name value ... | variable\_list}

The SETVAR command takes pairs of arguments, which may be preceded by an optional type-specifier (**-TYPE** type).

By default, the type of each variable specified in the SETVAR command depends on the type of each input value. You can, however, use the **-TYPE** argument to specify the individual type(s) of the variables.

Note: if a value is not of the type specified by the **-TYPE** argument, an error will result.

For more information on variables, refer to the *DOMAIN System User's Guide*.

## ARGUMENTS

variable\_list  
(optional)

Specify the names of the variables that receive the input values.

Default if omitted: must specify **-TYPE** (below).

## OPTIONS

**-TYPE** type var\_name...

Specify the type of the input value(s) that can be assigned to the particular variable name(s). Multiple variable names are permitted, separated by blanks. Once you specify a type in a particular SETVAR command, SETVAR assigns that type to all subsequent variable names, until you change the type specification. Valid types are:

STR[ING]	character strings
INT[EGER]	integer numbers
BOOL[EAN]	Boolean values
ENV[IRONMENT]	environment variables
ANY	any type (the default)

If the type of the input value does not match the type specified for that variable name, SETVAR issues an error and asks you enter another input value. Use **-TYPE ANY** to restore the Shell to its default state. In this case, it determines the proper variable type automatically.

Specifying **-TYPE ENV** var\_name causes the variable to become an environment variable. Environment variables are of primary concern to DOMAIN/IX users; please consult the DOMAIN/IX documentation for details about their usage.

## EXAMPLES

In the first example, we will create several variables using SETVAR and then list them using the LVAR command.

```
$ setvar i 1 # an integer
$ setvar b true # a boolean
$ setvar s1 string # a string
$ setvar -type string s2 true # a boolean forced into a string
$ lvar i b s1 s2
integer i = 1
boolean b = true
string s1 = string
string s2 = true
```

In this example, we will set several variables of different types and then list them.

```
$ setvar -type int i1 1 i2 2 -type str s1 3 s2 4 -type any i3 1
$ lvar i1 i2 s1 s2 i3
integer i1 = 1
integer i2 = 2
string s1 = 3
string s2 = 4
integer i3 = 1
```

The following is an error message example. In this case, we are trying to set an integer to a string.

```
$ setvar -type int z ten
?(sh) semantic error - 'ten' is not an integer.
```

## SH (SHELL)

**SH (SHELL) -- Invoke a Shell (command line interpreter).**

### FORMAT

**SH [options] [pathname [arg ...]]**

SH is the command line interpreter. It reads lines from standard input or from shell scripts; interprets them, finding the commands to which they refer and the arguments for the commands; and then invokes the commands. These commands provide all the traditional features of most computing systems: copying and deleting files, compiling and binding user programs, displaying system status, etc. Generally speaking, they do NOT create or control processes, windows, or other components of the display screen. (Those functions are provided by the Display Manager.) For general information about the Shell, see Chapter 3. The *DOMAIN System User's Guide* also includes information on the Shell, particularly the method that it uses to process input and the programming features available in scripts.

Normally, at least one process is running the Shell all the time. When you give the command SH, you generate a separate, subordinate Shell running within the current process. This Shell can carry on separate operations, and can execute special programs and scripts containing command lines. One of these scripts may be a startup script; see the -START option below for more information.

Note: The SH command does not create a new process, only a subordinate Shell running in the current process. To start a process (which may then run a Shell or any other program), use the DM command CP (CREATE\_PROCESS).

### ARGUMENTS

**pathname**  
(optional)

Specify file containing a Shell script to be executed. Each line in the file will be interpreted as a Shell command.

Default if omitted: read standard input.

**args**  
(optional)

Specify any arguments to be passed to the program in file 'pathname'. Arguments are substituted for ^n expressions in the program: arg1 for ^1, arg2 for ^2, etc. ^\* can be used to specify all of the arguments at once. (See the *DOMAIN System User's Guide* for details on passing arguments to Shell commands.) See example 1 below.

Default if omitted: no arguments passed.

### OPTIONS

Default options are indicated by "(D)."

**-B[ON]**

Send the output of a background process (created with the & parsing operator) to the display. The output of the background process is displayed in the transcript pad of the Shell where it

was invoked. If you do not specify **-B**, the output of the background process is sent to `/DEV/NULL`.

**-BOFF** (D) Do not display output from a background process.

**-NB[ON]** (D) Same as **-BOFF**.

**-C[OMMAND] arg1 ...**

Execute the following argument(s) as a Shell command, exactly as if it had been read as an input line. If any argument contains explicit blanks, enclose it in quotes. The Shell passes all text following **-C** to 'arg1' as arguments, so if you want to specify other options to the SH command itself, they must precede **-C**.

If 'arg1' is the name of a Shell script, note that the script creates a new Shell level for execution (just as if you had invoked it at the \$ prompt). Thus activities in the script that are level-dependent (such as assigning values to Shell variables) do NOT propagate upward when the script exits. This is in contrast to the **-START** option and the Shell command **SOURCE**, which execute scripts at the current Shell level.

**-E[ON]** Enable evaluation of variables outside of expressions. If **-E** is specified, the Shell always evaluates variables, regardless of the context in which they appear. If **-E** is not specified, variables are evaluated only inside variable expression delimiters, ((expression)); otherwise, the Shell treats the `^var_name` expressions as strings and they are not evaluated.

**-EOFF** (D) Evaluate variables only inside expressions.

**-NE[ON]** (D) Same as **-EOFF**.

**-F[IRST]** Do not exit after executing the command given by the **-C** option. This option is valid only if **-C** has been specified, and must precede **-C** on the command line.

**-I[NTER]** Behave as though input is being entered interactively: prompt for each input line, and do not exit on errors or quit faults (DQ or CTRL/Q from keyboard). Normally, the Shell only executes interactively if its input comes from a pad or SIO line. Use of this option forces prompting.

**-S[CRIP]T** (D) Behave as though executing a Shell script: do not prompt and abort on error. A Shell normally will not quit; any error or quit command is assumed to apply only to the last command given to the Shell.

**-NI[NTER]** (D) Same as **-S**.

**-N[EXECUTE]** Interpret each command line only; suppress execution.

**-EX[ECUTE]** (D) Interpret each command line and execute it.

## SH (SHELL)

- P[ROMPT]1 prompt\_string**  
Define the prompt string for the Shell created with SH.
- P[ROMPT]2 subprompt\_string**  
Define the subprompt string for the Shell created with SH. (The subprompt appears when you continue a Shell command over more than one line).
- START [file]**  
Execute the specified script after the Shell is created. If 'file' is not specified, the Shell searches the login home directory for a file called USER\_DATA/SH/STARTUP and executes it if it exists. No error occurs if that file does not exist.
- Note that the script is executed at the current Shell level, so that level-dependent activities (such as assigning values to Shell variables) persist when the script exits. This is in contrast to the -C option, which executes scripts at the next lower Shell level.
- This option is a default if SH is the first program invoked in a new process (i.e., CP /COM/SH). It is not a default at any other time (i.e., when you type SH at the dollar sign or call it from a script).
- NSTART**  
Disable startup file execution.
- V[ON]**  
Display each line of text in the transcript pad as it is read by the Shell program.
- VOFF** (D)  
Disable input verification.
- NV[ON]** (D)  
Same as -VOFF.
- X[ON]**  
Display each command in the transcript pad immediately before execution. Each command is given in full, with its complete pathname and with the values of arguments inserted.
- XOFF** (D)  
Disable input examination.
- NX[ON]** (D)  
Same as -XOFF.

### EXAMPLES

1. \$ sh program-name arg1 arg2 ...  
The Shell executes the commands in the file 'program-name', and substitutes the arguments ('argn') for character sequences ^n in the program file.
2. \$ sh -n my\_script  
Interpret each line in 'my\_script', but do not execute anything.

## SUMMARY OF INTERNAL SHELL COMMANDS

The Shell has four types of commands:

## External Commands

These are programs that reside on your disk. They are invoked when you type in their pathname or, if their directories are included in your command search rules, when you type their leafname.

## Internal Commands

These are built-in Shell commands (see below). The Shell always looks for these first.

## Control Structures

These are programming constructs that allow you to control the flow of control in a Shell script. Note: Since these structures are legal anywhere on the command line, you must enclose them in quotes when using the HELP command (i.e., HELP 'IF').

## Expressions

These are delimited by '()' and ')'. Inside of these double parentheses you can set variables, compare values and perform other standard integer, string or boolean operations. The assignment operation (VARIABLE := VALUE) is a special case that does not have to be enclosed in double parentheses.

Any of these commands can have their output redirected or may be invoked in the background using the Shells parsing operators (>, >>, >?, >>? <, <<, <?, <<?, |, &...) See the *DOMAIN System User's Guide* for details.

*Internal Commands*

Flags VON, VOFF, XON, XOFF, BON, BOFF, EON, EOFF

Variables READC, READ, READLN, EXISTVAR, LVAR, DLVAR, SETVAR, EXPORT

## Control Structures

IF, WHILE, SELECT, FOR  
EQS, EXISTF, RETURN, EXIT, NEXT, SOURCE, SET, ABTSEV,  
NOT

Miscellaneous ARGS, CSR, RDYM, HLPVER, INLIB, UMASK

Expressions TRUE, FALSE  
:=, OR, AND, =, <, >, <=, >=, <>, +, =, \*, /, MOD, \*\*, (, ), NOT

SH8 (INVOKE\_8-BIT\_SHELL)

SH8 (INVOKE\_8-BIT\_SHELL) -- Invoke 8-bit shell.

## FORMAT

SH8 [font]

The **sh8** program provides a shell with support for output of 8-bit characters. The standard output stream is extended through the use of a type manager to interpret characters above ASCII 128 as referring to characters in an additional font file. A new shell is then invoked by **sh8** which uses this stream. The value of the SHELL environment variable is used to determine which shell to invoke.

In order for **sh8** to work, there must exist in /sys/dm/fonts two font files, name (ASCII 32 to 126 decimal), name.a (ASCII 160 to 254). The name.b file (ASCII 128 to 159) is optional, as it is used to print out the control characters in the high 128 range (ASCII 128 - 159). If the program has no need to print these characters, the name.b file need not exist.

## ARGUMENTS

font

(optional)

The **sh8** default font is courier12. Thus, if **sh8** is invoked with no arguments, the files courier12, courier12.a, and (optionally) courier12.b must exist in /sys/dm/fonts.

Optionally, a font name may be given as the first (and only) argument to **sh8**.

The command **sh8 helvetica16** would use /sys/dm/fonts/helvetica16 as the low 128 characters, and /sys/dm/fonts/helvetica16.a as the high 128 characters.



## SHUTSPM

**SHUTSPM -- Shut down SPM on a node.**

### FORMAT

#### SHUTSPM

When the SPM runs in place of the DM, it waits on the eventcount file 'NODE\_DATA/SPMSHUT\_EC'. SHUTSPM advances this eventcount, causing the SPM to perform an orderly shutdown of the node.

To shut down the SPM with SHUTSPM, create a remote process (via the CRP command) on the target node and type 'SHUTSPM'.

Normally, only system administrators may shut down the SPM using this command. This is because SPM creates the 'NODE\_DATA/SPMSHUT\_EC' file with the following ACL (provided the default file ACL for 'NODE\_DATA' gives all rights to %.%.%.%):

Subject ID	Access Rights
%.sys_admin.%	pgndwrx
%.%.%.%	---d-r-

This ACL limits SHUTSPM shutdown to accounts with the 'sys\_admin' project name, but permits any account to delete the SPMSHUT\_EC file whenever SPM is not using it. If, however, the default file ACL for 'NODE\_DATA' has been changed, SPM creates the eventcount file using that default ACL. Note that an SID must have at least 'r' and 'w' rights to shutdown SPM.

If the SPMSHUT\_EC file already exists when SPM starts up, SPM does not change its ACL.

To prevent SPM from responding to the SHUTSPM command, add the following line to the 'NODE\_DATA/STARTUP.SPM' file:

```
NO_SHUTSPM
```

### EXAMPLES

```
$ crp -on 1fb -login sys_admin      Create remote process on server node 1fb  
                                     and log in with the system administrator  
                                     account.  
  
$ shutspm                            Shut down the SPM on server node 1fb.
```

**SIGP (SIGNAL\_PROCESS) -- Signal a Process.**

## FORMAT

**SIGP [process\_name ...] [options]**

SIGP causes a quit or stop fault in a process. This is particularly useful for stopping background processes such as those created by the CPO (CREATE\_PROCESS\_ONLY) and CPS (CREATE\_PROCESS\_SERVER) Display Manager commands.

You may discover which processes are currently active by using the PST (PROCESS\_STATUS) command.

## ARGUMENTS

**process\_name**  
(optional)

Specify name of process to be signalled. Multiple process names and wildcarding are permitted.

Default if omitted: -UID required (below)

## OPTIONS

Default options are indicated by "(D)."

- Q[UIT] (D)** Cause a quit fault in the process (like the Display Manager command DQ (CTRL/Q)). Executing programs halt, but the process remains active.
- S[TOP]** Ask the entire process to stop cleanly (closing streams, etc.).
- B[LAST]** Stop the process in the nucleus (don't go to user mode). This brings everything to a halt without letting the system attempt to clean up.
- UID high low**
- UID high.low** Stop the process with the given UID. "high" and "low" indicate the two halves of the UID.
- L** List processes signaled.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

1. 

```
$ sigp process_7 -quit
$
```

Generate a quit fault in process\_7, which will halt the program currently running there, but leave process\_7 itself active.

## SIGP (SIGNAL\_PROCESS)

```
2. $ sigp process_7 -stop -L   Stop process_7 completely.  
   "process_7" stopped.  
   $
```

SIORF (SIO\_RECEIVE\_FILE) -- Receive a file from a remote host.

## FORMAT

SIORF [options] [pathname ...] [\*]

SIORF accepts remote host transmissions from the appropriate SIO line, decodes them according to the proper protocol, and writes them to the file you specify.

Arguments and options may appear in any order and are processed and take effect as encountered. This means options must be specified *before* the file(s) for which they are intended.

You do not need to use the TCTL command to set the sync and insync parameters of the SIO line when receiving a non-ASCII file. SIOTF and SIORF recognize the types of the files being transferred and set these parameters correctly.

The transmission protocols used by SIORF are described in the SIOTF (SIO\_TRANSMIT\_FILE) command description.

## ARGUMENTS

**pathname**  
(optional)

Specify name of file to receive the transmission. If you omit the filename, SIORF waits for the host to specify a receive file. (If you want the transmission written to standard output, use the "\*" option.) SIORF terminates when it receives an end-of-transmission (EOT) signal, unless you include the -F option.

Default if omitted: see above

## OPTIONS

- L n            Specify SIO line being used for the transmission. The default SIO line is 1.
- N            Select the Nibble protocol. (See "Protocols" following the SIOTF (SIO\_TRANSMIT\_FILE) command description.)
- F            Cause SIORF to monitor the SIO line for host transmissions until it receives an error message over the SIO line or CTRL-Q from the node. When you include this option, EOT does not cause SIORF to terminate.
- OBJ         OBSOLETE OPTION. At SR9.5, SIORF automatically detects binary objects and receives them properly. Prior configuration of the SIO line (via the TCTL command) is no longer necessary.
- R            Replace file(s) if they already exist.
- AE          Abort on error. Otherwise, transmission continues until EOT.

## SIORF (SIO\_RECEIVE\_FILE)

- X host\_file** Request a remote host file to be transmitted. This presumes a host counterpart of SIOTF (SIO\_TRANSMIT\_FILE) is active.
- \*** Receive transmission to standard output. This option is valid only if the 'pathname' argument is omitted.

### EXAMPLES

1. `$ siorf -L 2 -r prog.bin` Create (or replace) a binary file PROG.BIN with the data received over SIO line 2; presumably being sent by an SIOTF counterpart.
2. `$ siorf -r -f` Receive files over SIO line 1 whose names are specified by the transmission side (host or DOMAIN node). Existing files are replaced if needed. SIORF remains active "forever" until CTRL/Q or error occurs.
3. `$ siorf -x ask_file /eng/new_copy` Request file ASK\_FILE to be sent over SIO line 1 and write data received to /ENG/NEW\_COPY. Presumes the other side is running SIOTF or equivalent in "forever" mode.

**SIO TF (SIO \_ TRANSMIT \_ FILE) -- Transmit a file to a remote host.**

## FORMAT

**SIO TF [options] [pathname ...] [\*]**

SIO TF sends the DOMAIN file(s) you specify to a remote computer ("host") using the appropriate SIO line and protocol.

Arguments and options may appear in any order and are processed and take effect as encountered. This means options must be specified *before* the file(s) for which they are intended.

You do not need to use the TCTL command to set the sync and insync parameters of the SIO line when receiving a non-ASCII file. SIO TF and SIO RF recognize the types of the files being transferred and set these parameters correctly.

## ARGUMENTS

**pathname**  
**(optional)**

Specify name of file to be transmitted. If you wish to transmit data from standard input, use the "\*" option.

Default if omitted: must use "\*"

## OPTIONS

**-L n**

Specify SIO line to be used for transmission. The default SIO line is 1.

**-N**

Select the Nibble protocol. (See the "Protocols" section below.)

**-F**

Cause SIO TF to continue monitoring the SIO line for transmission requests from the remote host rather than terminating when transmission is complete.

**-OBJ**

OBSOLETE OPTION. At SR9.5, SIO TF automatically detects binary objects and transmits them properly. Prior configuration of the SIO line (via the TCTL command) is no longer necessary.

**-AE**

Abort on error rather than attempting to continue.

**-X host\_file**

Pass a filename to the remote host. The host can use this name for the next file it receives from the node. This presumes a host counterpart to SIO RF (SIO \_ RECEIVE \_ FILE) is active.

**\***

Read from standard input and send standard input to the remote host. Signal end of data with an end-of-file (CTRL/Z).

**EXAMPLES**

- |                                      |  |
|--------------------------------------|--|
| 1. \$ siotf -f                       | Wait for file requests over SIO line 1 and transmit them.  |
| 2. \$ siotf -L 2 prog1.bin prog2.bin | Transmit file PROG1.BIN, then transmit file PROG2.BIN over SIO line 2.   |
| 3. \$ siotf -x tell_file /eng/notes  | Send the name 'tell_file', then transmit the file /ENG/NOTES. Presumably the receiving side is in "forever" mode (-F) and thus waiting for instructions. |

*PROTOCOLS*

To permit binary and ASCII file transmissions, we have implemented two protocols: Plain and Nibbled.

*Plain*

Plain protocol is the default. It assumes that the host operating system can transmit and receive all 256 bit patterns, so there is no need to use escapes or to nibble at the ASCII or binary files. Even if the host can handle only the ASCII character set, you should use the plain protocol for transmitting ASCII files. The format of this protocol is:

STX type COUNT...data...CHECKSUM CR

where:

STX is the standard ASCII STX (02).

type is a small ASCII letter that identifies the record type as follows:

a ACK    d DATA    e EOF    h HELLO    g ANSWER\_HELLO    n NAK  
p PARTIAL    t TYPE    x NAME    z EOT    ? ERROR MESSAGE

COUNT is the number of data bytes in the record (not to exceed 255), nibbled and transmitted as two ASCII bytes (@ and the capital letters A through O).

CHECKSUM is a 1-byte calculated checksum, nibbled and transmitted as two ASCII bytes.

CR is a standard ASCII carriage return.

The "t" and "p" types are provided primarily for file transfers occurring between DOMAIN nodes. "t" informs SIORF of the type of file being transmitted. In this case, the "data" field consists of a single character that identifies the type of file as follows:

"u": UASC file (normal ASCII text file)  
 "o": DOMAIN object file  
 "m": non-streams file, accessed though the mapping primitives  
 "r": streams record file

If you are transmitting a streams record file (type "r" above), the protocol now offers the "p" message type, which SIOTF uses to transmit partial records to SIORF. SIOTF can transmit at most 255 bytes at a time. If a record is larger than 255 bytes, transmission occurs in 255-byte pieces; all but the last piece will have the "p" type message. SIOTF transmits the last piece using the normal "d" type message so that SIORF will recognize it as the end of the record.

#### *Nibbled*

If the host cannot send or receive anything but ASCII characters, use Nibbled protocol to transmit non-ASCII data. Transmitted records use a record format identical to that of Plain protocol, except that ~S replaces the STX byte. For SIOTF, each byte from the file is nibbled into two ASCII characters (@ or A through O). For SIORF, the low four bits of each two bytes received are concatenated; this protocol checks the ASCII range of the received bytes. A byte out of range causes SIORF to send the host an NAK signal. A byte out of range in five consecutive records causes SIORF to issue an error message, and terminate. The count field of nibbled records contains the original count (i.e., the number data bytes before nibbling). To select the Nibbled protocol, use the -N option with SIORF or SIOTF.

When you execute SIORF, it issues the HELLO record to signal that it is there, and to clear any transmission that may have preceded your command. It expects to receive the ANSWER\_HELLO response. SIOTF also does this before it begins transmitting records.

SIORF acknowledges each remote host transmission. SIOTF waits for the host to acknowledge each transmission. These acknowledgements have the format:

STX a CR            (or)            STX n CR

STX is either STX or ~S, and a and n are the small letters a (ACK) and n (NAK). The programs recover from a NAK by retransmitting the record in question. After ten consecutive unsuccessful retries, the programs will issue an error message and abort. All messages must be acknowledged, including error messages.

END-OF-FILE is signalled by a record of this format:

STX e CR

where e must be the small letter e. Host programs should acknowledge the EOF signal.

END-OF-TRANSMISSION is signalled by a record of this format:

STX z CR

where z must be the small letter z. Host programs should ACK the EOT signal. If the programs do not receive transmissions or ACKs for 60 seconds, they issue time-out error messages and terminate.

*NOTES*

SIOTF opens a stream to its SIO line in COOKed mode, SIORF opens the stream in RAW mode. Both programs synchronize with host XON/XOFF (CTRL/Q, CTRL/S) signals.

If you specify a DOMAIN file that cannot be opened, the programs issue an error message. If a file specified by a record received from the host cannot be opened (or created), the programs will issue an error message, and transmit an error message to the host. However, they will continue processing their parameters or (if you specified -F) waiting for host requests.

SIOTF does not transmit EOT if you specify -F. SIORF will not terminate at EOT if you specify -F. If you omit -F, SIORF will wait until it receives an EOT signal from the host, or times out.

The programs accept type "?" error messages instead of ACK or NAK signals. The programs display the error messages, and terminate (even if you specified -F). If SIORF gets an error message while receiving a file, it aborts. If you included -F, the programs try to remain active as long as possible.

Model programs to serve as the host-side counterparts to SIORF and SIOTF have been supplied in /SYS/SOURCE/EMT. These are models in FORTRAN and in Pascal. The FORTRAN subroutines that need to be modified for host-specific use are in HOST\_MODEL\_SUBS1.FTN. The Pascal procedures to be modified are clearly marked in the Pascal model programs. For a particular host environment, you may also need to modify other areas of these models.

**SOURCE** -- Execute a Shell script at the current Shell level.

#### FORMAT

**SOURCE** *script\_name* [*arg1...*]

SOURCE allows you to execute a Shell script at the current Shell level. When you type

```
$ MY_SCRIPT arg1
```

your script runs in a new (subordinate) Shell level. This means that all variables are now defined at a new level; that your script can't delete or otherwise effect variables at the level above; that state settings like VON/BON/EON that get set in the script vanish when the script finishes, and so forth.

On the other hand, typing

```
$ SOURCE MY_SCRIPT arg1
```

executes the script at the current Shell level, just as though you had typed the contents of MY\_SCRIPT into the process input window (and filled in the command line arguments yourself). If the script says 'von' then VON will be set after the script exits. If it defines a variable, that variable will still be defined, etc.

#### ARGUMENTS

**script\_name**  
(required)

Specify the name of the script to be executed.

**arg1...**  
(optional)

Specify any arguments to be passed to the script.

Default if omitted: no arguments passed.

SRF (SORT\_FILE)

SRF (SORT\_FILE) -- Sort and/or merge text files.

## FORMAT

SRF [options] [pathname ...]

SRF sorts lines of all the named files together and writes the result on the standard output.

The sort key is an entire line. Default ordering is alphabetic by characters as they are represented in ASCII format (digits, then uppercase characters, then lowercase characters, then special characters).

## ARGUMENTS

<b>pathname</b> (optional)	Specify file(s) to be sorted. Multiple pathnames are permitted. Default if omitted: read standard input
-------------------------------	--

## OPTIONS

*Sort Key Control: one of the following*

- B Omit leading blanks from keys.
- S n Sort based on the subfield starting in column n. If this option is omitted, sorting starts in column one.
- F Fold all letters to a single case.

*Input Character Control: either of the following*

- D Use 'dictionary' order: only letters, digits and blanks are significant in comparisons. Special characters (punctuation, control characters, etc.) are ignored.
- I Ignore all nonprinting, nonblank characters.

*Sort Mode Control: one of the following*

- M Merge only, the input files are already sorted.
- R Reverse the sense of the sort; list output entries in reverse order.

## EXAMPLES

1. `$ ls -c -dtm | srf` List contents of current working directory in order of date last modified.

2. `$ ld -c -dtm | srf -r`

List most recently changed files first.

3. `$ ld -c -bl | srf -r`

List files by size with largest files first.

STCODE (STATUS\_CODE)

STCODE (STATUS\_CODE) -- Translate status code value to text message.

## FORMAT

STCODE hex\_stat\_code

STCODE prints the text message associated with a hexadecimal status code. This command is useful when a user program produces a hexadecimal status code instead of the textual message.

STCODE processes pre-defined status codes. No provision is currently made to add user-defined status codes to the error text database.

## ARGUMENTS

hex\_stat\_code  
(required) Specify hexadecimal status code to be translated.

## EXAMPLES

```
$ stcode 8001  
disk not ready (from OS / disk manager)
```

**SUBS (SUBSYSTEM) -- Set or display subsystem attributes.**

## FORMAT

**SUBS object [subs\_name] [options]**

SUBS is used to set or show protected subsystem attributes. When setting subsystem attributes, you must be running in that subsystem.

See the ENSUBS (ENTER\_SUBSYSTEM) command for details on entering a subsystem.

## ARGUMENTS

### object

(required)

Specify pathname of an object. The function of the object (either a protected file or a managing program) is determined by options described below.

### subs\_name

(optional)

Specify name of a subsystem. The Shell will search the directory /SYS/SUBSYS for the specified subsystem. If this argument is specified, the attributes of the named subsystem will be set as directed by the options described below.

Default if omitted: display attributes of 'object.'

## OPTIONS

### -DATA

Set or display the name of the subsystem which manages 'object.'

### -MGR

Set or display the name of the subsystem for which 'object' is a manager. 'Object' must be an executable file (i.e., a program).

### -UP

Increase the privilege level of a process running in a subsystem so that it can directly access the objects it owns.

### -DOWN

Decrease the privilege level of a process; opposite of '-UP.'

### -L

List subsystem attributes and/or manager fields. This is the default action if 'subs\_name' is not specified.

### -BR

Display only the name of the subsystem. Not valid if attributes are being set.

## EXAMPLES

The following example illustrates the use of protected subsystems. First we show a Pascal source program written to "manage" the subsystem. (The calls issued to /SYS/INS/ACLM.INS.PAS to enable proper subsystem ACL checking are documented in the *DOMAIN System Call Reference*.) Following that is a Shell script that installs the subsystem using the CRSUBS, ENSUBS, and SUBS commands.

## SUBS (SUBSYSTEM)

### *Pascal Source Manager*

```
{ pse --- protected subsystems example program }
```

```
{ usage:    pse pse_file out_file
```

```
  where:
```

```
    pse_file      protected object owned by 'PS_EXAMPLE' subsystem
    out_file      output file
```

The 'PSE' program is used to extract the protected data from objects owned by the 'PS\_EXAMPLE' subsystem and put them in an output file. As a trivial example, the protected data has a sequence number in the first 8 columns of each line, which is not logically part of the data, but which can be imagined to be important to the integrity of the data. Extracting the data removes the sequence number and copies the rest of the line to the output file. If this were a real application, it might also format and/or select the data sent to the output file.

```
}
```

```
program pse;
```

```
%include '/sys/ins/base.ins.pas';
%include '/sys/ins/streams.ins.pas';
%include '/sys/ins/error.ins.pas';
%include '/sys/ins/pgm.ins.pas';
%include '/sys/ins/aclm.ins.pas';
```

```
type
```

```
  buf_t = array[1..128] of char;
```

```
var
```

```
  istrid: stream_id_t;           { input stream ID }
  ostrid: stream_id_t;           { output stream ID }
  arg:    name_$pname_t;         { command line argument }
  alen:   integer;               { length of command line argument }
  st:     status_$t;             { status code }
  sk:     stream_$sk_t;          { stream seek key }
  buf:    buf_t;                 { I/O buffer }
  bp:     ^buf_t;                { pointer to same }
  blen:   integer32;             { length of I/O buffer }
```

```
begin
```

```
  { get input file name }
  alen := pgm_$get_arg(1, arg, st, sizeof(arg));
  if (st.code <> 0) then begin
    writeln('input file name missing. ');
    error_$print(st);
    pgm_$set_severity(pgm_$error);
    pgm_$exit
  end;

  { open input file; must increase privilege to access
    my own protected file... }
  aclm_$up;                               { get more privilege }
  stream_$open(arg, alen, stream_$read,
    stream_$no_conc_write, istrid, st);
  aclm_$down;                               { decrease privilege }
  if (st.code <> 0) then begin
    writeln('Can''t open input file. ');
    error_$print_name(st, arg, alen);
```

```

    pgm_$set_severity(pgm_$error);
    pgm_$exit
end;

{ get output file name }
alen := pgm_$get_arg(2, arg, st, sizeof(arg));
if (st.code <> 0) then begin
    writeln('output file name missing. ');
    error_$print(st);
    pgm_$set_severity(pgm_$error);
    pgm_$exit
end;

{ create output file; DO NOT increase privilege: it would
  be an error to write on one of my own protected objects
  here -- I want an ordinary file }
stream_$create(arg, alen, stream_$overwrite,
  stream_$no_conc_write, ostrid, st);
if (st.code <> 0) then begin
    writeln('Can''t create output file. ');
    error_$print_name(st, arg, alen);
    pgm_$set_severity(pgm_$error);
    pgm_$exit
end;

{ now just copy the file... a real program would be more
  complicated here. }
repeat
    { read a record... }
    aclm_$up;
    stream_$get_rec(istrid, addr(buf), 128, bp, blen, sk, st);
    aclm_$down;
    if st.code <> 0 then          { error or EOF }
        exit;
    { write a record, stripping off the sequence number.
      Notice I did NOT make a check to see that the length
      of the record was greater than 8 characters... I am
      confident that the rest of the subsystem correctly
      maintains sequence numbers, and that the protected
      subsystem mechanism makes sure that only the subsystem
      can operate on the data. }
    stream_$put_rec(ostrid, addr(bp^ [9]), blen-8, sk, st);
until st.code <> 0;

{ check that we stopped because of EOF }
if (st.subsys = stream_$subs) and then
    (st.code = stream_$end_of_file) then
    pgm_$exit;

{ not EOF --- a real error of some sort, then }
writeln('I/O error: ');
error_$print(st);
pgm_$set_severity(pgm_$error);
pgm_$exit
end.

```

## SUBS (SUBSYSTEM)

### *Shell Script*

```
# create a protected subsystem
crsubs ps_example
# create some data to be protected --- normally a special create
# operation would be used that guarantees data integrity
catf >ps_data <<!
12345678this is some protected data
12345679next record of protected data
!
ensubs ps_example -v <<!           # enter the new subsystem in a shell
subs pse ps_example -mgr          # make pse a manager of 'PS_EXAMPLE'
subs ps_data ps_example -data     # protect the data
edacl ps_data -d % -a %.backup r  # now can only get at data from within
subs -up
cpf ps_data ps_data2 -subs        # make a copy of the data
subs -down
!
pse ps_data out_file              # run PSE to extract the protected data
catf out_file                      # see the protected data
# now see how it fails if I try to make the output file a
# protected object of the 'PS_EXAMPLE' subsystem...
pse ps_data ps_data2              # try to clobber ps_data2
```

**TB (TRACEBACK) -- Print traceback after a fault.**

## FORMAT

**TB [process\_id] [options]**

TB prints a traceback after a fault. The traceback lists the names of the routines leading from the fault back to the main program, and includes the line number at which each routine was called. Up to 32 levels of calls can appear. If the sequence exceeds 32 calls, the first 16 and last 16 are shown in the traceback.

**NOTE:** There is a homonymous DM command: TB -- To bottom of window. See the TB command description in the DM chapter for details.

## ARGUMENTS

**process\_id**  
(optional)

Specify the ID of the process for which the traceback is desired. This may be the process name (displayed by the PST (PROCESS\_STATUS) command), the process UID, or the UNIX PID (from PST -UN).

Default if omitted: perform traceback for the current process.

## OPTIONS

The following options are meaningful only when used for cross-process tracebacks (i.e., tracing faults in some processes other than the current one).

**-ARGS** Print the arguments to each call mentioned in the traceback.  
**NOTE:** At SR9.5, this option functions only for Pascal and FORTRAN calls. It does not report arguments for C calls.

**-A[LL\_FRAMES]** Print all frames back to the root of the stack (i.e., through the current program and any calling programs) rather than stopping at the current program level.

**-R[EPEAT] [n]** Repeat the traceback every 'n' seconds. Repeat every 5 seconds if 'n' is omitted.

## EXAMPLES

1. In the following example, an overflow error occurred in a floating-point multiplication at line 25 of the routine named EXTRAPOLATE. The routine EXTRAPOLATE was called at line 192 of the routine REFLECTION, which in turn was called at line 19 of the main program.

## TB (TRACEBACK)

```
$ tb
overflow in multiply (from library/floating point)
In routine "EXTRAPOLATE" line 25
Called from "REFLECTION" line 192
Called from "$MAIN" line 19
$
```

2. Trace a fault in the process named 'mail', showing call arguments.

```
$ tb mail -args
In System Service "ec2_$wait"
Arguments unavailable
Called from "INPAD_$GET" line 152
  1. 00938A66 -> 00938ABC ....
  2. 00A53A1C -> 00000000 ....
  3. 00987664 -> 20750A2D u.-
  4. 000125B0 -> 000003E8 ....
  5. 0003907C -> 00000000 ....
Called from "STREAM_$GET_REC" line 149
  1. 00987AA0 -> 00000001 ....
  2. 00987650 -> 00987664 ..vd
  3. 000125B0 -> 000003E8 ....
  4. 00987A4C -> 00987A70 ..zp
  5. 00987A50 -> 000390E8 ....
  6. 00987A54 -> 00987638 ..v8
  7. 0003907C -> 00000000 ....
Called from "GET_COMMAND" line 1754
  1. 00987AA0 -> 00000001 ....
  2. 00987AA2 -> 00010002 ....
  3. 00039010 -> 3A000000 ....
  4. 000397DA -> 00010000 ....
  5. 000390E8 -> 0092576E ..Wn
  6. 0003907C -> 00000000 ....
Called from "EXEC_STREAM" line 2413
  1. 0000
  2. 0001
  3. 0002
  4. 0003
  5. 0003907C -> 00000000 ....
Called from "MAIL" line 3146
Arguments unavailable
$
```

3. Trace a fault in the process named 'mail' as far down the stack as possible.

```
$ tb mail -a
In System Service "ec2_$wait"
Called from "INPAD_$GET" line 152
Called from "STREAM_$GET_REC" line 149
Called from "GET_COMMAND" line 1754
Called from "EXEC_STREAM" line 2413
Called from "MAIL" line 3146
Called from "<Invoke Interlude>"
Called from "PGM_$LOAD_RUN" line 451
Called from "PGM_$INVOKE_UID" line 214
Called from "PM_$INIT" line 725
$
```

**TCTL (TERMINAL\_CONTROL) -- Set or display SIO line characteristics.**

## FORMAT

**TCTL [options]**

TCTL sets or displays SIO line characteristics, which control how hardware and software connected to those lines should behave. For example, if you wish to allow a dumb terminal to dial into a node and communicate meaningfully with a Shell, you must properly configure the SIO line that the terminal will use so that the node will understand the terminal's signals. Thus TCTL controls the transmission speed (baud rate) that connected terminals must use, and which characters typed on those terminals delete characters or lines.

## OPTIONS

If no options are specified, the current settings of the SIO lines are displayed.

- DEFAULT**                      Set all settable options to their default values. This allows you to quickly reset values to known states.
- LINE n**                        Specify the SIO line to be affected by subsequent options on this command line. 'n' is an integer in the range 0-3. The default SIO line is line 1 or standard input (if standard input is directed to an SIO line).
- SPEED baud**                    Set the speed of the line, for both input and output. The possible baud rates are: 50, 75, 110, 134, 150, 300, 600, 1200, 2000, 2400, 3600, 4800, 7200, 9600, 19200. The initial setting is 9600 baud. Note that 3600 baud is not supported on DN3xx systems. Speeds for partner line(s) may occasionally need to be forced: see -FORCE below.
- FORCE**                         Valid only if -SPEED is also specified. This option forces the speed of the line specified by -LINE to be set to the correct speed (specified by -SPEED). If the line has a "partner line" that is currently set to some other (incompatible) speed, -FORCE will reset the partner line's speed to 9600 baud. See example 4 below. For more information about partner lines, see the SIO\_\$CONTROL description in *Programming with General System Calls*.
- NLD [n]**                        Set NEWLINE delay. This is the number of milliseconds required following the output of a line feed (NEWLINE). If 'n' is omitted or not set, 20 milliseconds is the default.
- ERASE char**                    Set the erase character. This option is valid only when data is

## TCTL (TERMINAL \_ CONTROL)

being passed to the SIO line in "cooked" mode. 'char' may be any character or a one-byte hexadecimal value. Some characters may require quoting in the Shell. The erase character is initially set to BACKSPACE (08 hex).

- KILL char**            Set the kill character. This option is valid only when data is being passed to the SIO line in "cooked" mode. The kill character is initially set to CTRL/X.
- EOF char**            Set the end-of file character. The EOF character is initially set to CTRL/Z.
- QUITCHR char**        Set the quit character. The quit character is initially set to CTRL/].
- INTCHR char**         Set the interrupt character. This is used primarily by DOMAIN/IX. The interrupt character is initially CTRL/C.
- SUSPCHR char**        Set the suspend character. This is used primarily by DOMAIN/IX. The suspend character is initially CTRL/P.
- [NO]RAW**             Turn raw mode on or off. In raw mode, full 8-bit bytes are transmitted in both directions, without any interpretation. The initial setting is -NORAW.
- [NO]ECHO**            Turn the echoing of input characters over the SIO line on or off. The initial setting is ECHO.
- [NO]SYNC**            The terminal normally sends CTRL/S (XON) when its input buffer begins to fill, and CTRL/Q (XOFF) when the buffer begins to empty, to synchronize it with a high-speed data source. This option enables or disables that behavior (it is initially enabled). -SYNC implies -NORTS \_ENABLE.
- [NO]CVT \_NL**         Enable or disable conversion of LF to CR-LF on output. CVT \_NL causes NEWLINES (LF) to be transmitted as CR-LF sequences. This option is valid only when data is being passed to the SIO line in "cooked" mode. The initial setting is -NOCVT \_NL. NOTE: EMT always puts the SIO line in "raw" mode, so -CVT \_NL has no effect in that instance. Use the **OUTTERM** command within EMT.
- [NO]CVTRAW \_NL**     Similar to -CVT \_NL, but applies only to raw mode.
- [NO]INSYNC**         Enable or disable reacting to CTRL/S and CTRL/Q when received by node. -INSYNC causes transmissions to halt when CTRL/S is received and resume when CTRL/Q is received. The initial setting is -NOINSYNC.

- PARITY state**      Select parity checking state. Valid states are:
- |      |                                |
|------|--------------------------------|
| NONE | don't send or check parity bit |
| EVEN | send and check even parity     |
| ODD  | send and check odd parity      |
- The initial state is NONE.
- BPC n**      Set number of bits per character. 'n' is an integer in the range 5-8. The initial number of bits per character is 8.
- STOP n**      Set number of stop bits. 'n' may be 1, 1.5, or 2. The initial number of stop bits is 1.
- [NO]QUIT**      Enable/disable quits for the current process. The initial setting is -NOQUIT.
- [NO]INT**      Enable/disable interrupts for the current process. The initial setting is -NOINT.
- [NO]SUSP**      Enable/disable suspend faults for the current process. The initial setting is -NOSUSP.
- [NO]RTS**      Enable/disable the request-to-send line. The initial setting is -RTS. Note that you may NOT use this option if -RTS\_ENABLE is specified.
- [NO]DTR**      Enable/disable the data-terminal-ready line. The initial setting is -DTR. Note that -DTR is not valid if -LINE 3 is specified.
- [NO]DCD\_ENABLE**      Enable/disable standard handling of carrier detect. The initial setting is -NODCD\_ENABLE.
- [NO]CTS\_ENABLE**      Enable/disable standard handling of clear-to-send. The initial setting is -NOCTS\_ENABLE.
- [NO]RTS\_ENABLE**      Enable/disable RTS flow control. The initial setting is -NORTS\_ENABLE. Enable implies -NOSYNC.
- [NO]BP\_ENABLE**      Enable/disable processing of bit-pad input (from a graphics tablet) on the SIO line. When enabled, data received on this line is not delivered through STREAM\_\$GET\_REC, but is accumulated by the interrupt routine, and passed to the display driver a point at a time, much as with the touchpad. This processing has the additional property that subsequent points within +/-1 in both the X and Y dimensions are ignored. The initial setting is -NOBP\_ENABLE.

## TCTL (TERMINAL\_CONTROL)

### -ERROR state

Select error reporting state. Valid states are:

```
[NO]FRAMING    enable/disable reported framing errors
[NO]PARITY     enable/disable reported parity errors
[NO]DCD_CHANGE enable/disable report on DCD line
[NO]CTS_CHANGE enable/disable report on CTS line
```

Only FRAMING is initially enabled.

### EXAMPLES

- ```
$ tctl
```

 Display current settings.  
Status of Line 1:  
Erase (character delete) character: 08 (hex)  
Kill (line delete) character: 18 (hex)  
End of file character: 1A (hex)  
Quit character: 1D (hex)  
Interrupt character: 03 (hex)  
Suspend character: 10 (hex)  
New line delay: 0  
Speed: 9600  
Raw: FALSE, Echo: TRUE, Cvt\_NL: TRUE  
CvtRaw\_NL: FALSE, Host\_Sync: TRUE, Input\_Sync: FALSE  
RTS: TRUE, DTR: TRUE, DCD: FALSE  
CTS: FALSE, Quit\_Enable: FALSE, Int\_Enable: FALSE  
Susp\_Enable: FALSE, DCD\_Enable: FALSE, CTS\_enable: FALSE  
BP\_enable: FALSE RTS\_enable: FALSE  
Eight bits per character, Parity: None, One stop bit  
Errors enabled: FRAMING
- ```
$ tctl -line 2 -quitchar OFE -insync -speed 300
```

 Set quit character to hex FE, enable input synchronization, set speed to 300 baud on SIO line 2.
- ```
$ tctl -parity odd -quitchar '#' -kill !
```

 Set parity to odd, quit character to # (quoted because # normally begins a comment in the Shell), and kill character to ! on line 1.
- ```
$ tctl -line 2 -speed 50
```

?(tctl) Speed requested is incompatible with current speed of partner line 1. Resubmit command with -FORCE if permissible to reset partner line to 9600 baud.  

```
$ tctl -line 2 -speed 50 -force
```

?(tctl) Warning: Speed of partner line has been reset to 9600 baud.

**TEE -- Copy input to output and to named files.**

## FORMAT

**TEE** *pathname* ...

TEE copies its standard input to standard output and to the named files. It is useful for saving the data being transmitted through a pipeline.

## ARGUMENTS

**pathname**

(required)

Specify name of file to receive output. Multiple pathnames are permitted.

## EXAMPLES

```
$ FMT mary | TEE mary.clean | OS >mary.overstruck
```

This command line causes the file *mary* to be formatted with FMT. The formatted text is written to the file *mary.clean* and also piped to the OS command to produce overstruck output (for a line printer) redirected into the file *mary.overstruck*. Thus, you end up with two output files: one with ASCII carriage control (*mary.clean*) and one with FORTRAN carriage control (*mary.overstruck*).

## TLC (TRANSLITERATE\_ CHARACTERS)

TLC (TRANSLITERATE\_ CHARACTERS) -- Replace characters.

### FORMAT

**TLC from-chars [to-chars]**

TLC copies standard input to standard output, substituting or deleting selected characters. Each input character found in 'from-chars' is replaced by the corresponding character of 'to-chars'.

TLC differs from CHPAT (CHANGE\_ PATTERN) in that it deals only with single characters or ranges of characters, whereas CHPAT deals with character strings. For example,

```
$ tlc xy yx
```

changes all x's into y's and all y's into x's, whereas

```
$ chpat xy yx
```

changes all the patterns "xy" into "yx".

### ARGUMENTS

**from-chars**

**(required)**

Specify existing character(s) to be replaced. You may specify a range of characters by separating the extremes with a dash. For example, a-z stands for the list of lowercase letters. 'from-chars' may contain a maximum of 100 characters.

**to-chars**

**(optional)**

Specify replacement characters. You may specify a range of characters by separating the extremes with a dash. For example, a-z stands for the list of lowercase letters. 'to-chars' may contain a maximum of 100 characters.

If 'from-chars' and 'to-chars' contain an equal number of characters, TLC translates the first character in 'from-chars' to the first character in 'to-chars', and so forth.

If 'from-chars' contains more characters than 'to-chars', TLC repeats the last character in 'to-chars' until 'to-chars' is as long as 'from-chars'. However, in the output, adjacent repetitions of the last character appear as one character. (See example 2 below.)

If 'to-chars' contains more characters than 'from-chars', the extra characters are ignored.

Default if omitted: delete all occurrences of characters in the 'from-chars' list.

## EXAMPLES

The following examples show TLC's operation using standard input and output. The first line following the command line is an echo of standard input. The next line is the TLC results, then another line of input, then more results, and so forth.

```
1. $ TLC te zq
   Now is the time
   Now is zhq zimq
   *** EOF ***
   $
```

```
2. $ TLC abc zq
   Now is the time for all good men and boys to come to the aid
   Now is the time for zll good men znd qoys to come to the zid
   abcaccbaa
   zqzqzz
   aaaaaa
   zzzzzz
   bbbbbb
   q
   ccccc
   q
   *** EOF ***
```

Note that multiple occurrences of "a" are replaced by "z" one for one, but multiple occurrences of "b" and "c" are replaced with a single "q", since the 'from-char' list is longer than the 'to-char' list.

```
3. TLC A-Z a-z <mary.caps >mary.lc
```

This command changes all uppercase letters in the input file "mary.caps" to lowercase and writes the results to the file "mary.lc". Lowercase characters already in "mary.caps" remain unchanged.

TPM (TOUCH\_PAD\_MODE)

TPM (TOUCH\_PAD\_MODE) -- Set/display touchpad and mouse characteristics.

## FORMAT

### TPM [options]

TPM allows you to define characteristics for the touchpad and mouse. The touchpad operates in one of three modes: absolute, relative, and absolute/relative. The mode of operation establishes how movements of your finger on the touchpad affect the position of the cursor on the screen. The three modes differ primarily in how the cursor moves when you lift your finger from the touchpad and then replace it. The subsections below describe the three operational modes, as well as the other options.

The mouse operates in relative mode only.

## OPTIONS

If no options are specified, TPM displays the current touchpad characteristics.

Default options are indicated by "(D)."

- A (D) Select absolute mode.
- R Select relative mode.
- AR Select absolute/relative mode.
- RERANGE Set prescaling factors for touchpad data.
- S x y Set scaling factors for x and y. Values must be in raster units, and can range from 1 to 1024. The default scaling factors are 799 for x and 1023 for y (portrait displays); and 1023 for x and 799 for y (landscape displays).
- O x y Set x and y as the origin for absolute mode. Values must be in raster units, and can range from 0 to 1023. The default origin is 0,0.
- H n Set the hysteresis box size. The value must be in raster units, and can range from 0 to 1023. The default is 5.

### *Absolute Mode*

In absolute mode, using the default scale and origin, the touchpad approximates the screen, so that the top left edge of the touchpad represents cursor positions at the top left edge of the screen. Absolute mode is the default setting. When you place your finger on the touchpad, the cursor jumps to a corresponding position on the screen. Moving your finger across the touchpad moves the cursor across the screen in the same direction.

For example, moving your finger from the top of the touchpad to the bottom moves the cursor

from top to bottom on the screen. If you lift your finger from the touchpad, and later touch the pad again, the cursor jumps to a new position on the screen corresponding to the new finger position.

Absolute mode has no meaning if you are using a mouse.

### *Relative Mode*

In relative mode, cursor movements correspond only to finger movements across the touchpad. The cursor does *not* move when you first place your finger on the touchpad. This differs from absolute mode, where the cursor jumps to a new position when you lift your finger and then replace it. In effect, relative mode causes the touchpad to correspond to different areas of the screen, relative to the current cursor position.

This is the only meaningful mode for a mouse: all movement begins from the current cursor position.

Relative mode is typically used with scale factors less than the defaults. Smaller scale factors mean that the touchpad maps to a smaller area of the screen. For example, scale factors of 200 by 256 specify one-sixteenth of a portrait display's screen area. With small scale factors, relative mode allows fine resolution of the cursor position within a small area.

To reach distant areas on the screen, you can use several "strokes" on the touchpad or mouse, each stroke moving the cursor closer to its final destination. To assist you in making large movements in relative mode without having to use too many strokes, the speed of cursor movement is artificially accelerated in relation to the speed of finger or mouse movement. Thus, a quick motion will move the cursor farther than a slow, deliberate motion which covers the same distance.

### *Absolute/Relative Mode*

Absolute/relative mode is a combination of absolute and relative modes. It has no meaning for the mouse. In this mode, the first position of your finger on the touchpad establishes the first position of the cursor, as in absolute mode. Moving your finger across the touchpad moves the cursor across the screen. As in relative mode, the scale is typically smaller than the whole screen.

Unlike absolute and relative modes, however, the effect of lifting your finger from the touchpad depends on how long you break contact. If you lift and replace your finger quickly -- within a half second -- the cursor does not move, and the effect is the same as relative mode. If you break contact for more than a half second, however, the cursor jumps to a new absolute position when you put your finger on the touchpad again.

Absolute/relative mode is useful for "jumping" the cursor from one place to another, then carefully positioning it in the new area. For example, this mode is commonly used to move the cursor in a jump from one window to another, and then point to a character in the second window.

## TPM (TOUCH\_PAD\_MODE)

### *Prescaling the Touchpad*

Raw touchpad data vary slightly from one touchpad to another. Prescaling is, in essence, calibration of the touchpad. Every time you start the node, the touchpad manager prescales the data to determine an exact range for the device.

To prescale, the touchpad manager observes the first thousand points of touchpad data (about 30 seconds of use). During this time, you should try to touch all four edges of the touchpad to ensure that the observed data constitute an accurate sample. Based on the observed data, the touchpad manager computes a prescaling factor which, when applied to the data, brings all points into the range from  $-0.05$  to  $1.05$ . This range corresponds to the edges of the screen, plus an overlap of 5%, when multiplied by the default scaling factors. Because of the overlap, you need not touch the internal frame (under the conductive material) to move the cursor to the edge of the screen.

The `-RERANGE` option invokes prescaling. This option is useful if the first 30 seconds of use did not include the entire range of the touchpad. It is also handy if you change keyboards on a node, and therefore need to reset the prescaling factors without restarting the node.

### *Scale Factors*

The touchpad manager translates, or *scales*, the data into raster units, which the Display Manager understands. Scale factors, specified with the `-S` option, are applied to the prescaled touchpad data to translate it to raster units for the Display Manager.

The scale factors are multiplied by the prescaled data. The default scale factors are 800 for x and 1024 for y (portrait displays); and 1024 for x and 800 for y (landscape displays). Applying these factors to prescaled data results in numbers from approximately 0 to 799 (for x) and 0 to 1023 (for y) for portrait displays, and vice versa for landscape displays. (Note that the prescaled data allow a 5% overlap, as mentioned in the preceding subsection.)

The default scale factors provide for touchpad data corresponding to the whole screen. Smaller scale factors reduce the area to which the touchpad maps, thereby allowing you to more finely tune the cursor position. This also applies to mouse movement, allowing changes in the apparent motion sensitivity of the device.

### *Setting the Origin*

The origin is the point denoted by the upper left corner of the touchpad, in absolute and absolute/relative mode. In relative mode, the origin has no meaning. By default, the touchpad origin corresponds to the upper left corner of the screen, that is, the point 0,0 in raster units. By changing the origin, you can use the touchpad (in absolute mode) to correspond to a portion of the screen.

This feature is useful for applications that need to move the cursor within a fixed window, rather than across the whole screen. For example, a program that displays a menu in one window might set the origin to the upper left corner of the menu window. Consequently, the touchpad maps onto the menu window instead of the entire screen.

### *Hysteresis*

The hysteresis value defines the dimensions of a "box" around your finger position on the touchpad or the current position of the mouse. Movement within the box does not change the position of the cursor on the screen.

Specify the hysteresis value in raster units. The touchpad manager compares the value to the difference between the current and previous finger positions on the touchpad or the current and previous positions of the mouse. If the difference is less than the hysteresis value, the cursor does not move. If the difference is greater than the hysteresis value, the hysteresis value is subtracted from the difference and the cursor moves the resulting distance. The default hysteresis value is five.

### **EXAMPLES**

\$ TPM	Display current characteristics
Mode: absolute	
Xscale: 1024, Yscale: 800	
Hysteresis: 5	
Origin: 0, 0	
\$ TPM -ar -s 400 512	Set characteristics to absolute/relative mode with half the default scaling sensitivity (portrait display).

## TZ (TIMEZONE)

**TZ (TIMEZONE) -- Set or display system time zone.**

### FORMAT

```
TZ [ tz_name | utc_delta [new_tz] ]
```

TZ sets the system time zone to a known time zone or to an offset from Coordinate Universal Time (UTC).

To set the actual time registered by the nodes's internal clock, use the CALENDAR command. See the CALENDAR command description for more information.

### ARGUMENTS

If no arguments are specified, TZ displays the current setting.

**tz\_name**  
(optional)

Specify new time zone. Valid names are:

<i>Name</i>	<i>Time Zone</i>
EDT	Eastern Daylight Time
EST	Eastern Standard Time
CDT	Central Daylight Time
CST	Central Standard Time
MDT	Mountain Daylight Time
MST	Mountain Standard Time
PDT	Pacific Daylight Time
PST	Pacific Standard Time
GMT	Greenwich Mean Time
UTC	Coordinated Universal Time

Default if omitted: use 'utc\_delta' argument

**utc\_delta**  
(optional)

Specify positive or negative offset from UTC. The plus sign is optional for positive offsets. Format for offset is hh:mm (e.g., -10:00 for ten hours earlier than (west of) Coordinated Universal Time). Only whole or half hour offsets may be specified. Other fractional offsets produce an error message.

Default if omitted: use 'tz\_name' argument

**new\_tz**  
(optional)

Specify new time zone name to be assigned to the zone indicated by the 'utc\_delta' argument. Use this argument to create time zones that are not included in the list above.

Default if omitted: no name assigned

## EXAMPLES

```
$ TZ
  Timezone:      EST
  Delta from UTC: -5:00
```

Display current time zone.

```
$ TZ pdt
```

Set time zone to Pacific Daylight Time

```
$ TZ 4:30 gst
```

Create (and set) a time zone named GST that is four and a half hours later than (east of) Coordinated Universal Time.

UCTNODE (UNCATALOG\_NODE)

UCTNODE (UNCATALOG\_NODE) -- Uncatalog a node.

## FORMAT

**UCTNODE** *pathname* ... [*options*]

UCTNODE removes the specified entry directory name from the local copy of the network root directory. After the name is removed, objects cataloged under that node's entry directory are no longer accessible to you or other nodes on the network.

If you use the **-ROOT** option, the nodename is also removed from the network's replicated root directory.

Node entry directories are created with the **CTNODE** (CATALOG\_NODE) command.

## ARGUMENTS

<b>pathname</b> (required)	Specify node entry directory name to be uncataloged. Multiple pathnames and wildcarding are permitted.
-------------------------------	--

## OPTIONS

<b>-L</b>	List directory names as they are uncataloged.
<b>-ROOT</b>	Uncatalog the node in the network root as well as in the the local root directory.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

<pre>\$ uctnode als_node</pre>	Uncatalog the node with the entry directory name specified.
--------------------------------	---

**UCTOB (UNCATALOG\_OBJECT) -- Uncatalog the specified pathname, without deleting the associated object.**

## FORMAT

**UCTOB** pathname ... [options]

UCTOB removes the specified pathname from the name space. The object associated with the pathname is not affected. This command is primarily intended for system-level debugging use.

## ARGUMENTS

**pathname**  
(required)

Specify name of object to be uncataloged. The object itself is not affected. Multiple objects and wildcarding are permitted.

## OPTIONS

**-BR**

Suppress listing of names and UIDs of objects as they are uncataloged. These are reported unless this option is specified.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
$ uctob testfile                               Uncatalog "testfile".
"testfile" uid is 16791COC.40000074.
$
```

ULKOB (UNLOCK\_OBJECT)

ULKOB (UNLOCK\_OBJECT) -- Unlock an object.

## FORMAT

ULKOB [pathname ...] [options]

The ULKOB command unlocks objects residing on, or locked by processes running on, the current node. You may not unlock objects on remote nodes unless you locked them (see -F below). This command can be used when a program terminates abnormally, leaving objects locked, or to unlock objects previously locked with the LKOB (LOCK\_OBJECT) command.

To obtain a list of your node's locked objects, use the LLKOB (LIST\_LOCKED\_OBJECTS) command.

## ARGUMENTS

**pathname**  
(optional) Specify name of object to be unlocked. Multiple pathnames and wildcarding are permitted.  
Default if omitted: -U option must be specified

## OPTIONS

If no options are specified, the object is unlocked for all lock modes.

- R** Unlock an object that was locked for read mode; the lock must be owned by this process.
- W** Unlock an object that was locked for write mode; the lock must be owned by this process.
- I** Unlock an object that was locked for reading with intent to write; the lock must be owned by this process.
- F** Forcibly unlock an object. It may have been locked for any mode and the lock may be owned by any process. The object must reside on the current node, however, or must have been locked by the current node. In other words, you cannot unlock objects on a remote node unless you locked them.
- L** List the name of each object as it is unlocked.
- U uid ...** Specify the UID of the object(s) to unlock. Multiple UIDs are permitted. If the 'pathname argument is omitted, then this option is required.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ ULKOB mary -f                      Forcibly unlock the file "mary" for any mode.

\$ ULKOB -uid 1C1A9E2F.20000246 1C1A9E42.50000246  
                                    Unlock the two objects with the specified UIDs.

## UMASK

**UMASK -- Set DOMAIN/IX file-creation mode mask.**

### FORMAT

**UMASK [nnn]**

UMASK sets or displays your DOMAIN/IX file-creation mode mask. This is an internal Shell command.

### ARGUMENTS

**nnn**  
**(optional)**

Specify the read/write/execute permissions for *owner*, *group*, and *others*, respectively. The value of each specified octal digit is subtracted from the corresponding "digit" specified by the system for the creation of a file. Refer to the DOMAIN/IX description of **creat(2)** for more information about file creation. Also see **chmod(1)** and **umask(2)** for further discussion of file permissions.

Default if omitted: display current mask value.

### EXAMPLES

To remove write permission of the group and others, execute the following command.

```
$ umask 022
```

Files normally created with mode 777 become mode 755; files created with mode 666 become mode 644.

VCTL (VT100\_CONTROL) -- Set/display VT100 terminal characteristics.

## FORMAT

VCTL [options]

VCTL allows you to set or display information about how the VT100 terminal emulator driver handles input from the keyboard (for example, whether or not it echoes characters, or how it interprets key sequences typed at the keyboard).

This command is valid only if you have the VT100 terminal emulation software package running on your node. In addition, VCTL can only be run in a window where the VT100 emulator is already running.

## OPTIONS

If no options are specified, the current VT100 settings are displayed.

- DEFAULT           Set all options to their default values. This allows you to quickly reset values to known states.
- [NO]CVT\_IN\_NL    Convert a newline (linefeed) to a carriage return on input. The initial setting is -NOCVT\_IN\_LINE.
- [NO]CVT\_IN\_CR    Convert a carriage return to a newline on input. The initial setting is -CVT\_IN\_CR.
- [NO]CVT\_OUT\_NL   Convert a newline to carriage return, newline on output. The initial setting is -CVT\_OUT\_NL.
- [NO]CVT\_OUT\_CR   Convert a carriage return to a newline on output. The initial setting is -NOCVT\_OUT\_CR.
- [NO]ECHO          Turn the echoing of input characters on or off. The initial setting is ECHO.
- [NO]ECHO\_CTL     Turn the echoing of control characters (such as CTRL/Z) on or off. The initial setting is NOECHO\_CTL.
- [NO]ECHO\_ERASE   If ECHO is on, controls whether characters are visibly erased from the screen when the erase character is typed. The combination of ECHO and NOECHO\_ERASE causes the erase character to be echoed until all characters on a line are erased. The initial setting is -ECHO\_ERASE.

**-[NO]RAW**

If RAW is on, a program reading from the keyboard in the VT100 will receive each character as it is typed. If RAW is off, such a program will block until a full line has been typed. A full line is a sequence of characters ending in a newline character. In other words, in non-raw mode, a program blocks until a carriage return or line feed is typed.

**-[NO]ECHO\_KILL**

If ECHO is on, controls whether a line is visibly erased from the screen when the line kill character is typed. The combination of ECHO and NOECHO\_KILL causes the kill character to be echoed and a new line to be displayed. The initial setting is -ECHO\_KILL.

**-EOF char**

Set the end-of-file character. The EOF character is initially set to CTRL/Z.

**-ERASE char**

Set the erase character. This option is valid only when data is being passed to the terminal emulator in "cooked" mode. The 'char' can be any character or one-byte hexadecimal value. Some characters may require quoting in the Shell. The erase character is initially set to BACKSPACE (08 hex).

**-INTR char**

Set the interrupt character, which sends an interrupt fault to the process group of the terminal emulator. The interrupt character is initially set to CTRL/C.

**-KILL char**

Set the kill character. This option is valid only when data is being passed to the emulator in "cooked mode". The kill character is initially set to CTRL/X.

**-QUIT char**

Set the quit character. The quit character is initially set to CTRL/Q.

**-SUSP char**

Set the suspend character. The suspend character is initially set to hex FF, which is equivalent to its being disabled.

**-[NO]ENABLE\_SIGS**

If ENABLE\_SIGS is on then the fault-generating characters (interrupt, quit, suspend) have their special meaning. If ENABLE\_SIGS is off then these characters are not treated specially.

**EOL char**

Set the extra break character. The EOL character is initially set to hex FF, which is equivalent to its being disabled. If it is enabled, the EOL character behaves like <return> in that any program reading from the keyboard will immediately wake up and read whatever has been typed so far, including the EOL character itself.

## EXAMPLES

1. \$ vctl Display current settings.  
 Erase (character delete) character: "^H" (08 hex)  
 Kill (line delete) character: "^U" (15 hex)  
 End of file character: "^Z" (1A hex)  
 Interrupt character: "^C" (03 hex)  
 Quit character: "^Q" (11 hex)  
 Extra break character: FF (hex)  
 Suspend character: FF (hex)  
 Raw: FALSE, Echo: TRUE, Echo\_Erase: TRUE  
 Echo\_Kill: TRUE, Echo\_Ctl: FALSE, Cvt\_In\_CR: TRUE  
 Cvt\_In\_NL: FALSE, Cvt\_Out\_NL: TRUE, Cvt\_Out\_CR: FALSE  
 Enable\_Sigs: TRUE  
 \$
  
2. \$ vctl -quit OFE -cvt\_out\_cr Set quit character to hex FE,  
enable conversion of output  
newlines to carriage returns.



**VOFF -- Deactivate the Shell's -V flag.**

**FORMAT**

**VOFF**

VOFF turns off the Shell's -V (Verify) flag, which is turned on by the VON command or the -V option on the SH command line. When the flag is off, command lines are not displayed when they are read by the Shell. Verification is off by default.

VOFF requires no arguments or options.

VON

**VON -- Activate the Shell's -V flag.**

#### FORMAT

##### VON

VON turns on input verification. As commands are executed, or comments processed, they are written to the error output stream of the Shell. In Shell scripts, VON can be used to show the progress being made by the script.

If VON is turned on in a shell script, it remains on until that shell script exits, or until over-ridden by a VOFF in a nested shell script. When a shell script exits, the state of input verification is returned to the state in effect just before the script was invoked.

VON requires no arguments or options.

**VSIZE (VT100\_SIZE) -- Set/display VT100 window settings.**

## FORMAT

**VSIZE [options]**

The **VSIZE** command allows you to set the dimensions of the VT100 emulator window pane. This command is only valid from within the VT100 emulator (which is invoked with the **VT100** command); attempting to use it directly from the Shell causes an error.

## OPTIONS

If no options are specified, **VSIZE** displays the current window pane settings.

- L n** Specify the height of the window pane in lines. If this option is omitted, the height remains unchanged.
- C n** Specify the width of the window in columns. If this option is omitted, the width remains unchanged.
- STD** Set the height of the window to 24 lines and the width to 80 columns (same as saying **-l 24 -c 80**).

## EXAMPLES

```

$ vt100                Invoke VT100 emulator.
$ vsize                Display current settings.
Screen size is 18 lines by 70 columns.
$ vsize -c 60          Change the width.
Old screen size is 18 lines by 70 columns.
New screen size is 18 lines by 60 columns.
$ *** EOF ***         Exit the emulator and return
$                      to the Shell.
```

**VT100 -- VT100 terminal emulator.**

## FORMAT

**VT100 [options] [pathname [arg1 arg2 ...]]**

The VT100 command creates a window running the VT100 terminal emulator and starts up a Shell within the window. This command is valid only if you have the VT100 terminal emulation software package running on your node.

The VT100 terminal emulation package is intended for use with two types of programs. When used in conjunction with remote communications packages such as DOMAIN TCP/IP or X.25, the VT100 terminal emulator allows you to interact with the remote system as if you were logged into a VT100 connected to that system. Using the VT100 terminal emulator with programs that take advantage of VT100 special features allows you to run these programs on a DOMAIN node without having to tailor them to the DOMAIN environment.

The VT100 terminal emulation package consists of:

- The terminal emulation software, which performs the functions of a VT100 terminal, such as handling VT100-type escape sequences. The terminal emulator redirects the handling of keyboard input and screen output to stream manager operations. The terminal emulator is invoked within a DM window by the VT100 Shell command.
- The terminal emulator driver, which performs keyboard input functions such as erasing or echoing characters. The VCTL Shell command allows you to set and display the VT100 terminal characteristics controlled by the terminal emulator driver.

## ARGUMENTS

If any options are specified, they must precede the argument(s).

**pathname [arg1 arg2 ...]**

**(optional)**

Specify the name of a command or program for the Shell in the VT100 window to invoke. You must give the full pathname; for example, /com/ld. Arg1, arg2, ... are valid arguments to the selected command (or program): for example, /com/ld //my\_node/my\_home\_dir.

Default if omitted: invoke /com/sh

## OPTIONS

If any options are specified, they must precede the argument(s). Once VT100 is running, you may change the window size with the VSIZE (VT100\_SIZE) command.

**-STD**

Set up a VT100 window that is 24 lines by 80 columns (the standard size of a VT100 screen).

**-LINES n**                    Set up a VT100 window with the number of lines specified by 'n'. The number of lines cannot be greater than the number of lines in the DM window running the VT100 emulator.

**-COLUMNS n**                Set up a VT100 window with the number of columns specified by 'n'. The number of columns cannot exceed the number of columns of the DM window running the VT100 emulator.

## EXAMPLES

1. \$ VT100                    Create a window running the VT100 emulator and start a shell running within the window.
2. \$ VT100 /COM/TELNET hostname            Open a connection to the remote system specified by 'hostname' and create a window running the VT100 emulator.

### *VT100 Keyboard Layout*

The table below shows how the keys on a DOMAIN low-profile or 880 keyboard map to the keys of a VT100. This presupposes that you are running the VT100 Keyboard Emulation package on your node. Note that the VT100 definitions for the <F2>, <F3>, and <F7> keys supercede the usual EMT definitions for these keys.

<i>DOMAIN key</i>	<i>VT100 keypad</i>
<INS MODE>	<ESC>
<CHAR DEL>	<RUBOUT>
<F2>	<PF1>
<F3>	<PF2>
<F4>	<PF3>
<F5>	<PF4>
SHIFT/<F2>	<7>
SHIFT/<F3>	<8>
SHIFT/<F4>	<9>
SHIFT/<F5>	<->
CTRL/<F2>	<4>
CTRL/<F3>	<5>
CTRL/<F4>	<6>
CTRL/<F5>	<.>
<F6>	<1>
<F7>	<2>
SHIFT/<F6>	<3>
SHIFT/<F7>	<ENTER>
CTRL/<F6>	<0>
CTRL/<F7>	<.>

WBAK (WRITE\_BACKUP)

WBAK (WRITE\_BACKUP) -- Create a magnetic media backup file.

## FORMAT

WBAK pathname ... -F file\_no [options] [-]

WBAK writes one or more objects to a magnetic media backup file. These objects may be directory trees, files, or links. For each object, the information saved includes the name, object data, and attributes associated with the object, such as the access control list. This lets you reconstruct files, the directory tree, or any portion of the tree using the RBAK (READ\_BACKUP) command.

The WBAK and RBAK commands are intended both for disk backup and for interchanging information between separate DOMAIN installations. Use the RWMT (READ\_WRITE\_MAGTAPE) command to read and write magnetic media which are used for interchanging information with non-DOMAIN installations.

### *Tape Structure*

WBAK writes the contents of the objects you specify to a single "backup file". Note that a backup file may be an ANSI standard tape file or diskette, and may contain many DOMAIN files, directories, and links. A backup file is a logically (and, if contained on one physical volume, physically) contiguous area of magnetic media surrounded by ANSI "file header" and "end of file" labels. One physical backup volume may contain one or more backup files. A single backup file may, however, span multiple magnetic media volumes.

The collection of backup files on one or more associated physical magnetic media volumes is called a "file set". The first backup file on the first physical magnetic media volume in a file set is numbered "1". Subsequent backup files in this file set are numbered in ascending order from "2".

### *Backup Modes*

If you are backing up directory trees, WBAK can operate in one of three modes: "full" backup, "incremental" backup, or "dtm relative" backup. In full backup mode, all files, directories, and links are written to the backup file. When doing a full backup, objects in use do not get backed up. In incremental backup mode, all files are saved which were modified since the last full or incremental backup (when the backup history file was updated). In dtm relative mode, all files which were most recently modified either before or after the specified time are written to the tape.

### *Backup History*

WBAK records all times that a directory has been backed up in a file called BACKUP\_HISTORY. This file is updated in all directories named on the command line with the 'pathname' argument; it is not updated in directories contained within (subordinate to) those named directories. If no directory is named on the command line, then no BACKUP\_HISTORY file is made. The information written to this file includes the date and time of the backup (in Coordinate Universal Time (UTC)), the backup mode, and, if you have specified a dtm relative backup, the date and time to which the backup is relative.

WBAK uses this file in incremental backup mode to determine the date and time of the last full or incremental backup. This file is a standard text file and may be read in the same way as any text file; you should not, however, change it except possibly to delete old entries from the beginning of the file if it becomes too large. The automatic update of the history file can be suppressed by using the -NHIST command option.

#### *File Identification on Tape*

Associated with a backup file is a "file id" (-FID option). This is a 1 through 17 character user-assigned name which can be used in place of the file number to identify the backup file. This name is stored in the file header label and is printed (by default) by RBAK when the contents of a backup file are indexed (listed) or restored.

#### *Full Disk Backup*

An entire disk can be backed up by specifying the entry directory name as the pathname (example 2). It takes approximately 25 minutes to perform a full backup on a local 33 megabyte Winchester disk; 50 minutes for a remote disk.

#### *Backup Verification*

Use RBAK with the -INDEX option to list a single backup file. For an index of all backup files on one physical tape volume, use RWMT with the -INDEX option.

When using WBAK, please note the following:

- Directories must allow list access in order to be backed up.
- Files must allow read access in order to be backed up.
- Objects locked for writing by another process cannot be backed up.
- WBAK must be run on the node which is connected to the tape or floppy unit. You may accomplish this either by physically typing the WBAK command on the host node, or by running WBAK in a process on the host node created from your own remote node using the CRP CREATE\_PROCESS command.
- Only one tape unit can be connected to any node.
- There are no special tape mounting commands. Simply mount the tape on the tape drive and execute WBAK.

## ARGUMENTS

**pathname**  
(required)

Specify the name of the object to be written to backup media. This may be a directory, file, or link. If it is a file, then the file is written as specified. If it is a link, then the link is resolved and the resolution object is written to backup media. If it is a

## WBAK (WRITE\_BACKUP)

directory, all subordinate files and subdirectories in the tree are written. Note that the pathname specified reflects the way the directory is stored on the backup media, and that the same name must be used when reading files using pathnames in RBAK. Multiple pathnames and wildcarding are permitted. If you omit this argument, WBAK will prompt you for it. You may specify a hyphen (-) as an argument to direct WBAK to standard input for further arguments and options.

### OPTIONS

The -F option is required, as it specifies where on the backup media the new file is to be written. If you omit it, WBAK will prompt you for it.

Default options are indicated by "(D)."

#### *Tape File Identifiers*

**-FID file\_id** Specify a 1-17 character file ID to be written in the file header label for use when writing a file to a labeled volume. If this option is omitted, the file is not named and can only be restored by the file number.

**-F [position]** Specify the file position for the write operation. Valid values for 'position' are "CUR", "END", or a nonzero integer. A position of "CUR" specifies that the file should be written at the current position on the backup media; the media must have been previously written by WBAK and its position must not have been disturbed.

A position of "END" specifies that the file should be written at the end of the backup media file set. This causes WBAK to append the specified disk file ('pathname' argument) to the very end of the file set.

A position specified by a nonzero integer value causes the file to be written at that absolute position in the backup media volume. If multiple 'pathname' arguments are supplied, the value of 'position' is incremented by one after each file has been written.

The default value for 'position' is 1.

#### *Mode Control*

The object specified by the 'pathname' argument must be a directory for either -FULL or -INCR to have meaning.

**-FULL (D)** Specify a full backup; save all files in specified trees.

**-INCR** Specify an incremental backup; save files which were modified since the last backup recorded in the BACKUP\_HISTORY file stored in the 'pathname' directory.

**-AF dtm** Save all files modified after the given date and time; dtm is in the form "yy/mm/dd.hh:mm". The date defaults to today, and the time to midnight if either of those are omitted from dtm.

**-BEF dtm** Save all files last modified before the given date and time.

*Label Control*

**-WLA (D)** Write the backup media volume label if the backup file number is 1.

**-NWLA** Suppress writing of the backup media volume label.

**-OWN id** Specify backup media volume owner (1-14 character name). This option is only meaningful when used with the **-WLA** option.

**-VID vol\_id** Specify a 1-6 character volume ID for use when labeling a volume. This option is only meaningful when the backup file number is 1. The default volume ID is ' ' (blank).

**-SLA (D)** Display the label information written for this backup file on standard output.

**-NSLA** Suppress output of label information.

*Listing Control*

You may include the **-L** option, or any combination of **-LD**, **-LF**, AND **-LL**.

**-L** Write the names of all files, directories, and links saved to standard output.

**-LF** Write the names of all files saved to standard output.

**-LD** Write the names of all directories saved to standard output.

**-LL** Write the names of all links saved to standard output.

*Backup Device Control*

**-DEV d[unit]** Specify device type and unit number. 'd' must be either 'M' (for reel-to-reel magnetic tape), 'CT' (for cartridge tape), or 'F' (for floppy), depending on which drive is being used. 'unit' is an integer (0-3). Both are required for reel-to-reel tapes (i.e., **-DEV M2**). A unit number is NOT required for floppy disks and cartridge tapes (i.e., **-DEV F**). If this option is omitted, RBAK assumes device M0.

**CAUTION:** Floppy disk support for this command is limited. In particular, error detection during reads and writes is poor. DO NOT use this command with floppy disks when the data being placed on the floppy disks are critical and unrecoverable.

**-REO** Force previous backup media volume to be reopened, and suppress reading of backup media volume label. Use only when backup media has not been repositioned since last WBAK or RBAK.

## WBAK (WRITE\_BACKUP)

### *Special Cartridge Tape Control Options*

- RETEN** Retension the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.
- NRETEN (D)** Do not retension the cartridge tape.
- NO\_EOT** Suppress the writing of two tape marks at the end of the tape file, which are the standard signal for end of tape. The cartridge can't position between the two tapemarks to be ready for a successive call to WBAK (as it does on magtape), without rewinding the tape and searching forward, so this option speeds up multiple invocations of WBAK. *It SHOULD NOT be used on the LAST invocation of WBAK.* Also, '-F CUR' should be used on all WBAK invocations in a series except the first one.
- SYSBOOT** Permit use of a bootable tape that has a special boot program at the beginning. This option causes WBAK to skip over the first file on the tape. This option is only necessary when the first file on the tape is being written ('-F 1').

### *Miscellaneous Control Options*

- NHI** Suppress update of the backup history file.
- (hyphen)** Read standard input for further arguments or options; input is accepted until WBAK receives an EOF (CTRL/Z by default).
- PDTU** Preserves the last date/time-used information on objects. After each object is backed up on tape, the date/time-used information is reset to the value it had before the backup.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

## EXAMPLES

```
1. $ wbak //mask/wby -f 1 -af 81/11/19.12.00 -fid wby -L
```

This command writes the directory //MASK/WBY to tape. The directory is written out to tape file one, and the file ID "wby" is written to the file's label. Disk files from directory WBY are written to the tape only if they have been modified since noon on November 19, 1981. The label and the names of the files written to tape are printed to standard output.

When this command is executed, WBAK writes the following information to standard output:

```
Label:
  File number: 1
  File section: 1
  File ID: wby
  Date written: 1981/11/20 10:47:58 EST
```

Starting write:

```
(file) "//mask/wby/among" written
(file) "//mask/wby/school" written
(file) "//mask/wby/children" written
(file) "//mask/wby/backup_history" written
(dir) "//mask/wby/" written.
```

Write complete.

2. \$ wbak -f 1 -own "john doe" -vid "volbk2" -fid "node 27 backup" //goeey

This command backs up the entire contents of the node whose entry directory name is "goeey". Note that the file ID is specified as "node 27 backup" to make it easy to identify when you want to reload it, and that the command assigns volume and owner IDs.

When this command is executed, WBAK writes the following information to standard output:

```
Label:
Volume ID:      VOLBK2
Owner ID:       john doe
File number:    1
File section:   1
File ID:        n 27 backup
File written:   1983/02/17 18:00:39 EST
```

Starting write:

Write complete.

3. \$ wbak -f 1 -own "john doe" -vid "volbk1" ug/[a-f]?\*\_example -1

This command uses wildcards to match only those files in the "ug" subdirectory of the current working directory whose names begin with the letters "a" through "f" and end with "\_example".

When this command is executed, WBAK writes the following information to standard output:

```
Label:
Volume ID:      VOLBK1
Owner ID:       john doe
File number:    1
File section:   1
File ID:        (no ID specified)
File written:   1983/02/17 17:58:52 EST
```

Starting write:

```
(file) "ug/cmf_example" written.
(file) "ug/cmt_example" written.
(file) "ug/cpboot_example" written.
(file) "ug/cpf_example" written.
(file) "ug/cpt_example" written.
(file) "ug/fpat_example" written.
(file) "ug/fppmask_example" written.
```

## WBAK (WRITE\_BACKUP)

(file) "ug/fst\_example" written.

Write complete.

## DIAGNOSTICS

### *I/O Errors*

When WBAK has an I/O error, it attempts the operation again, for a total of five times. After the fourth retry fails, WBAK prints out an error message describing which type of error occurred. If the error was during an attempt to write to a tape, WBAK skips the tape block which caused the error, and tries to write the same data in the next block. Note that no data is lost, but RBAK will return an I/O error when it tries to read that block. If the write attempt again fails, after five tries, WBAK skips that block and tries the next. This process will continue for a total of twenty consecutive failed blocks, at which time WBAK aborts.

tape rewind error

An I/O error occurred.

tape write-filemark error

An I/O error occurred.

tape space-filemark error

An I/O error occurred.

tape space-record error

An I/O error occurred.

i/o recovery failed

An I/O error occurred and the tape drive could not reposition for another try.

tape i/o error

An I/O error occurred.

### *Operator Errors*

first label on volume is not VOL1 label

Expected a standard label, and did not find one.

label version number in VOL1 label is not "3"

The label format is incorrect.

a HDR1 label is missing where one is required

A file on the tape does not begin with the correct format.

wrong volume, file header is inconsistent with previous trailer

The wrong continuation tape was put on the drive. This is an operator error which can occur when a multi-tape volume is used.

magtape drive is offline

You have not put the drive on line.

tape is write-protected

The write enable ring is not on the tape.

**file not found**

The tape file specified was not found.

**invalid unit number**

Tape unit specified is not connected. Presently, only DEV 0 is supported.

**pbu is not present.**

No tape unit is connected to the node. WBAK can only be run on the node connected to the tape drive.

WD (WORKING\_DIRECTORY)

WD (WORKING\_DIRECTORY) -- Set or display the current working directory.

## FORMAT

WD [pathname]

WD sets the working directory for the current process to the specified directory. The working directory is where the system looks for objects when you don't explicitly specify a directory as a part of a pathname.

## ARGUMENTS

**pathname**  
(optional)

Specify new working directory. This may be a derived name, but must point to a directory or link to a directory. Specifying a file will cause an error. WD also accepts the command line parser arguments "-" and "\*". If you specify a hyphen (-), WD looks to standard input for the directory name. An asterisk (\*) followed by the name of a file directs WD to look inside that file for the new working directory name.

Default if omitted: display current working directory.

## EXAMPLES

- ```
$ wd //fred/jtj
$ wd
//fred/jtj
$ wd stuff/revised
$ wd
//fred/jtj/stuff/revised
```

Set new working directory.  
Display the new setting.  
Set working directory with derived name.  
Display the new setting.
- ```
$ wd -
//frodo/my_stuff
*** EOF ***
$
```

Direct input to standard input for new directory name.  
Signal end of input with CTRL/Z.
- ```
$ wd *newdir
$
```

Direct input to a file named "newdir" that holds the name of the new working directory.

**WHILE** -- Execute a **WHILE** loop.

**FORMAT**

**WHILE** condition **DO** command ... **ENDDO**

**WHILE** executes a command (or commands) as long as the results of a Boolean test are true. You can extend the **WHILE** command over several lines if you use it interactively or in a Shell script. When you use **WHILE** interactively, and extend the command over more than one line, the Shell prompts you for each new line of the command with the `$_` prompt.

**ARGUMENTS**

**condition**  
(required)

Specify a command or program to execute and test for truth, or specify a variable expression or Boolean variable to test for truth. "Truth" usually means that the command executes successfully (without any errors), or that a Shell variable expression or Boolean is "true". (Specifically, this argument is evaluated **TRUE** if it returns an abort severity level of 0 (zero).)

Refer to the *DOMAIN System User's Guide* for more information on Shell variables.

**command ...**  
(required)

Specify the command(s) or program(s) to execute if 'condition' returns **TRUE**.

**EXAMPLES**

```

1. $ eon
   $ K := 3                               (set var K = 3 )
   $ WHILE ((^K > 0))
   $_ DO args ((^k)); k := ^k - 1         (body of WHILE )
   $_ ENDDO
   3                                       (output of WHILE)
   2
   1
   $

```

XDMC (EXECUTE\_DM\_COMMAND)

XDMC (EXECUTE\_DM\_COMMAND) -- Execute a DM command from the Shell.

## FORMAT

XDMC **dm\_command** [**args...**]

XDMC allows you to invoke Display Manager commands from the command Shell or from within a Shell script. This is similar to pressing the <CMD> key on the keyboard and then typing the DM command in the DM input window, which is the usual way to perform DM operations.

## ARGUMENTS

**dm\_command**

**(required)**

Specify the Display Manager command to be executed. See Chapter 2 for DM command descriptions.

**args ...**

**(optional)**

Specify any arguments to be passed to the DM command. These are sent directly to the DM without further processing by the command Shell.

Default if omitted: no arguments passed

## EXAMPLES

1. \$ xdmc dq  
Cause the DM to send a quit fault to the current process. This is analagous to the SIGP (SIGNAL\_PROCESS) Shell command, with the following important difference. Whereas SIGP accepts an argument designating which process to fault, this example will send a fault to whatever process (if any) is pointed to by the current cursor position.
2. \$ xdmc cp /com/sh  
Cause the DM to create a new process and invoke the Shell. This is the same as pressing the <SHELL> key.

**XOFF -- Deactivate the Shell's -X flag.**

**FORMAT**

**XOFF**

XOFF turns off the Shell's -X (execution trace) flag, which is turned on by the XON command or by the -X option on the SH command. When the flag is off, command lines are not displayed before execution. The flag is off by default.

XOFF requires no arguments or options.

XON

**XON -- Activate the Shell's -X flag.**

## FORMAT

### XON

XON turns on execution tracing. Just before each command is executed, its full pathname and arguments are written to the error output stream of the Shell. In Shell scripts, XON can be used to show the progress being made by the script, and can help debug Shell scripts by showing the actual arguments being passed to commands, after all Shell processing on them is complete.

By default, execution tracing is off when a Shell is invoked.

If XON is turned on in a Shell script, it remains on until that Shell script exits, or until over-ridden by a XOFF in a nested Shell script. When a Shell script exits, the state of execution tracing is returned to the state in effect just before the script was invoked.

XON requires no arguments or options.

**XSUBS (EXECUTE\_SUBSYSTEM) -- Run Shell script subsystem manager.**

## FORMAT

**XSUBS** pathname [args...]

Once a protected subsystem, a subsystem manager(s), and a subsystem data object(s) exist, any user can execute the manager program. To run a binary manager program, you simply execute the program. To run a Shell script manager program, you must use the XSUBS command. Note that in order to see the name of a subsystem created on another node, you must copy the file /SYS/SYBSYS/Subsystem\_name to your node. If you do not copy this file, you can use the subsystem managers to operate on the objects, but when you ask to display the name of the subsystem, you will get an error message like the following:

```
$ subs //fred/jtj/com/top_secret
?(subs) Can't show subsystem manager type for "//fred/jtj/com/top_secret"
      - subsystem name not found (US/aclm)
$
```

## ARGUMENTS

**pathname**  
(required)            Specify Shell script containing the subsystem manager to be executed. Note that this script must contain the commands SUBS -UP and SUBS -DOWN in order to enter and exit the subsystem.

**args ...**  
(optional)            Specify arguments to be passed to the Shell script.

Default if omitted: no arguments passed

## EXAMPLES

Suppose you have an append-only list that you wish to protect. Anyone can read the list, and append to the list, but no one can overwrite previously existing contents. Assume that the subsystem 'append\_only' already exists. Then the 'APP' Shell script, which appends standard input to an append-only file, would look like this:

```
# APP --- append to an append_only file
SUBS -UP
CATF >>^1            # append to the file passed as first argument
SUBS -DOWN
```

To make APP a manager of the 'append\_only' subsystem, do

```
ENSUBS append_only            # enter subsystem
SUBS APP append_only -MGR
*** EOF ****
```

## XSUBS (EXECUTE\_SUBSYSTEM)

A run of APP would look like this:

```
XSUBS APP aofile          # execute APP on 'aofile'  
this is the stuff that is appended  
*** EOF ***
```

# Index

## A

AA 2-2  
Abort severity 4-2, 4-229  
Abort text search 2-3, 2-66  
ABRT 2-3  
Absolute mode 4-277  
Absolute/relative mode 4-278  
ABTSEV 4-2  
Access control 4-4, 4-88,  
4-238  
Access rights 4-92  
Accounts 4-85  
Acknowledge DM alarms 2-2,  
2-6  
ACL 4-4  
ACLs  
    display or copy 4-4  
    edit 4-88  
    explained 4-91  
    salvage 4-238  
AD 2-4  
Add to a window group 2-87  
Address space, list 4-160  
AL 2-5  
Alarm server 4-245  
Alarms  
    acknowledge 2-2  
    pop window 2-6  
    send 4-245  
Analyze network performance  
4-190, 4-193, 4-213  
AP 2-6  
AR 2-7  
ARCF 4-7  
Archive files 4-7  
ARGS 4-9  
Arrow keys  
    move down one line 2-4  
    move left one character  
    2-5  
    move right one character  
    2-7  
    move up one line 2-9  
    set scale factors 2-8  
AS 2-8  
ASCII carriage control 4-202,  
4-205  
Assign user input to variables  
4-225, 4-227, 4-228  
AU 2-9  
Autohold mode, window 2-81

## B

Background color, display  
2-10  
Background process output  
4-17, 4-18  
Backup history 4-294  
Badspots, described D-1  
BGC 2-10  
    See also INV  
BIND 4-10  
Bind object modules 4-10  
BLDT 4-16  
BOFF 4-17  
BON 4-18  
Boot Shell 2-36

## C

CALENDAR 4-19, A-1  
Cancel ECHOing 2-66  
Cancel text search 2-3, 2-66  
Carriage control 4-202, 4-205  
CASE 2-11  
    change in text 2-11,  
    4-275  
CASE construct 4-243  
Case sensitivity 2-63  
Catalog a node 4-55  
Catalog an object 4-58  
Catalog uncatalogued objects  
4-126  
Catenate files 4-20  
CATF 4-20  
CC 2-12  
CDM  
CE 2-13  
Change  
    case of text 2-11, 4-275  
    home directory 2-47, 4-21  
    icon into window 2-39  
    object name 4-22  
    password 2-46, 4-24  
    window into icon 2-39  
    window position 2-91, '  
    2-92  
    window size 2-85, 2-86  
Check for file existence 4-121  
Check internet router 4-230.1  
Check variable existence  
4-122

CHHDIR 4-21  
 CHN 4-22  
 CHPASS 4-24  
 CHPAT 4-25  
 Close a pad 2-82  
 Close a window 2-82  
 CMACCT 4-27.1  
 CMDF 2-14  
 CMF 4-28  
 CMPPO 4-29.1  
 CMS 2-15  
 CMSRF 4-30  
 CMT 4-31  
 Command line interpreter  
   4-250  
 Command line parser 3-4  
   standard command options  
   3-6  
 Command search rules 4-53  
 Compare  
   account files 4-27.1  
   directory trees 4-31  
   files 4-28, 4-30  
   PPO files 4-29.1  
   strings for equality 4-117  
 Conditional statement 4-155,  
   4-243  
 Continue a process 2-23  
 Conventions  
   in this manual 2  
 Convert file types 4-59  
 Coordinates  
   X/Y 1-2  
 Copy  
   directory tree 4-39  
   display image 2-97  
   entire display 4-38  
   file 4-34  
   link 4-36  
   system boot file 4-33  
   text to paste buffer 2-95  
   window 2-12  
 Count strings in a file 4-128  
 CP 2-16  
 CPB 2-18  
 CPBOOT 4-33  
 CPF 4-34  
 CPL 4-36  
 CPO 2-20  
 CPS 2-21  
 CPSCR 4-38  
 CPT 4-39  
 CPU time, show 4-224  
 CRD 4-42  
 Create  
   directory 4-42  
   edit pad 2-13  
   file (zero-length) 4-44  
   link 4-45  
   local registry 4-49  
   network registry 4-49  
   paste buffer 2-18  
   protected subsystem 4-51  
   read-only edit pad 2-22  
   remote process 4-46  
   type object 4-51.2  
   types 4-51.1  
   user account 4-85  
   user change request 4-52  
   window group 2-87  
   window to transcript pad  
   4-48  
 Create a process 2-16  
   background 2-20  
   server 2-21  
 CREFS 4-43  
 CRF 4-44  
 CRL 4-45  
 Cross-reference text strings  
   4-43  
 CRP 4-46  
 CRPAD 4-48  
 CRRGY 4-49  
 CRSUBS 4-51  
 CRTY 4-51.1  
 CRTYOBJ 4-51.2  
 CRUCR 4-52  
 CSR 4-53  
 CTNODE 4-55  
 CTOB 4-58  
 Current date and time 4-61  
 Cursor movement  
   move down one line 2-4  
   move left one character  
   2-5  
   move right one character  
   2-7  
   move up one line 2-9  
 Cut and paste  
   copy image 2-97  
   copy text 2-95  
   cut text 2-96  
   paste text 2-98  
 Cut text to paste buffer 2-96  
 CV 2-22  
 CVT\_REC\_UASC 4-59

## D

DATE 4-61  
DC 2-23  
DCALC 4-62  
DEBUG 4-65  
Debug high level programs  
4-65  
Declare keyboard type 2-43  
Define user names 4-102  
Defining keys 2-44  
Defining points and regions  
1-1  
Delete  
    character under cursor  
        2-28  
    directory 4-72  
    duplicate lines from file  
        4-69  
    file 4-70  
    link 4-71  
    preceding character 2-29  
    types 4-72.1  
    variables 4-73  
    window from window group  
        2-88  
Desk calculator functions 4-62  
Directory  
    copy 4-39  
    create 4-42  
    delete 4-72  
    list contents 4-167  
    naming 4-189  
    salvage 4-239  
    working 4-302  
Disks  
    dismount 4-74  
    initialize 4-157, D-1  
    mount 4-185  
    salvage 4-241, E-1  
Dismount a volume 4-74  
Display  
    refresh 2-59  
    set background color 2-10  
Display process status 4-76,  
4-217  
DLDUPL 4-69  
DLF 4-70  
DLL 4-71  
DLT 4-72  
DLTY 4-72.1  
DLVAR 4-73  
DM commands from a Shell  
4-304  
DM messages 2-49  
DMTVOL 4-74  
Documentation conventions 2

DQ 2-24  
DR 2-25  
DS 2-26  
DSPST 4-76  
Dumb terminal emulator  
4-108

## E

ECHO 2-27  
Echo command line arguments  
4-9  
ED 2-28, 4-78  
EDACCT 4-85  
EDACL 4-88  
EDFONT 4-96, C-1  
EDIP 4-96.1  
Edit  
    file (DM) 2-13  
    file (Shell) 4-78  
    font 4-96, 4-201, B-1,  
        C-1  
    IP\_HELPER 4-96.1  
    magtape descriptor file  
        4-97  
    network root directory  
        4-99  
    stream 4-105  
EDMTDESC 4-97  
EDNS 4-99  
EDPPO 4-102  
EDSTR 4-105  
EE 2-29  
EEF 2-30  
EI 2-31  
Elapsed time, show 4-224  
EM3270 4-107  
EMT 4-108  
Emulate 3270 terminal 4-107  
Emulate dumb terminal 4-108  
EN 2-32  
End-of-file marks 2-30  
ENSUBS 4-114  
Enter a protected subsystem  
4-114  
ENV 2-33  
Environment variables 2-33,  
4-124  
EOF marks 2-30  
EOFF 4-115  
EON 4-116  
EQS 4-117  
ER 2-34  
Erase marks 2-15  
ES 2-35

ESA 4-118  
EX 2-36  
Execute DM script 2-14  
EXFLD 4-119  
EXISTF 4-121  
EXISTVAR 4-122  
EXIT 4-123  
Exit DM to Boot Shell 2-36  
Exit from a loop 4-123  
Expand macro definitions  
4-181  
EXPORT 4-124  
External symbol address  
4-118  
Extract fields of data 4-119

## F

Fault status 4-147  
File existence test 4-121  
File length 4-128  
File protection 4-4, 4-88,  
4-238  
Find blocks of text 4-142  
Find spelling errors 4-145  
Find text strings 1-2, 4-139,  
4-142  
FIND\_ORPHANS 4-126  
FL 2-37  
FLEN 4-128  
Floating-point error mask  
4-144  
Floppy disks  
read backup 4-219  
write backup 4-294  
FMC 4-129  
FMT 4-130  
Fonts  
described B-1  
edit 4-96, 4-201, B-1,  
C-1  
load 2-37  
FOR 4-137  
FOR loop 4-137  
Format  
multiple columns 4-129  
text file 4-130  
FORTRAN carriage control  
4-202, 4-205  
FPAT 4-139  
FPATB 4-142  
FPPMASK 4-144

Free disk blocks, list 4-180  
FSERR 4-145  
FST 4-147

## G

GM 2-38  
Grow a window 2-85, 2-86

## H

HELP 4-148  
for Shell scripts 4-149  
Histogram, program counter  
4-151  
HLPVER 4-149  
Hold mode, window 2-89  
Home directory 4-21  
HPC 4-151  
Hysteresis 4-280

## I

ICON 2-39  
Icon default positions 2-40  
Icons  
create 2-39  
set default positions 2-40  
IDF 2-40  
IF 4-155  
Initialize a disk volume 4-157,  
D-1  
INLIB 4-156  
Insert characters 2-35  
Insert mode 2-31  
Install  
types 4-156.1, 4-156.2  
user libraries 4-156  
Interrupt a process 2-26  
INTM 4-156.1  
INTY 4-156.2  
INV 2-42  
See also BGC  
Invisible windows 2-90  
INVOL 4-157, D-1  
IP\_HELPER 4-96.1  
ITEST 4-157.1

## K

KBD 2-43  
KD 2-44  
Key names  
    standard 1-8  
Key naming conventions 1-8  
Keyboards  
    880 map 1-9  
    declare type 2-43  
    low-profile map 1-9  
Keys, defining 2-44

## L

L 2-46  
LAMF 4-158  
Laminate files 4-158  
Language level debugger 4-65  
LAS 4-160  
LBR 4-162  
LCM  
LCNET 4-164.1  
LCNODE 4-165  
LD 4-167  
Librarian utility 4-162  
Libraries, install user 4-156  
Line mode editor 4-78  
Link  
    copy 4-36  
    create 4-45  
    delete 4-71  
List  
    address space 4-160  
    connected networks  
        4-164.1  
    connected nodes 4-165  
    directory contents 4-167  
    installed types 4-176  
    locked objects 4-171  
    network registry sites  
        4-175  
    open streams 4-174  
    users logged in 4-177  
    variables 4-179  
    volume free space 4-180  
LKOB 4-170  
LLKOB 4-171  
LO 2-48  
Load a font 2-37  
Locate blocks of text 4-142  
Locate spelling errors 4-145  
Locate text strings 1-2,  
    4-139, 4-142

Lock an object 4-170  
Locked objects  
    list 4-171  
    lock 4-170  
    unlock 4-285  
Log in to a node 2-46  
Log in to a process 4-172  
Log off a node 2-48  
Log out 2-48  
Logical volumes, described  
    D-1  
Login 4-172  
    list users 4-177  
    to a node 2-46  
    to a process 4-172  
Loops  
    exit from 4-123  
    FOR 4-137  
    return to top 4-197  
    WHILE 4-303  
LOPSTR 4-174  
LRGY 4-175  
LTY 4-176  
LUSR 4-177  
LVAR 4-179  
LVOLFS 4-180

## M

MACRO 4-181  
Magnetic tape descriptor file  
    4-97  
Marks 2-25, 2-38, 2-57  
    erase 2-15  
    go to 2-38  
    place 2-25  
    replace 2-57  
Merge files 4-262  
Merge registries 4-184.1  
Messages, DM 2-49  
Mount a disk 4-185  
Mouse characteristics 4-277  
Move a file 4-187  
Move a pad 2-50, 2-51,  
    2-53, 2-54, 2-55  
Move a window 2-91, 2-92  
Move cursor  
    down one line 2-4  
    left one character 2-5  
    right one character 2-7  
    to bottom of window 2-67  
    to DM window 2-68  
    to end of line 2-76  
    to front of line 2-72  
    to input window 2-71

to next icon 2-75  
to next tab 2-69  
to next window 2-74  
to previous tab 2-70  
to previous window 2-73  
to top of window 2-78  
to window border 2-79  
up one line 2-9  
MRGRGY 4-184.1  
MSG 2-49  
MTVOL 4-185  
Multiple columns 4-129  
MVF 4-187

## N

Naming directory 4-189  
Naming server 4-99  
ND 4-189  
Negate a command or expression 4-198  
NETMAIN 4-190  
    log files 4-191, 4-192  
NETMAIN\_CHKLOG 4-191  
NETMAIN\_NOTE 4-192  
NETSTAT 4-193  
NETSVC 4-195  
Network maintenance 4-190, 4-191, 4-192  
    log files 4-191, 4-192  
Network registry 4-85  
    create 4-49  
    list sites 4-175  
    salvage 4-240  
Network services 4-195  
Network statistics 4-193, 4-213  
NEWLINE, insert 2-32  
NEXT 4-197  
Nibbled mode protocol 4-259  
Node clock 4-19, A-1  
Node specifications 3-2  
Nodes, list connected 4-165  
NOT 4-198  
NS\_HELPER 4-99

## O

Object types 4-199  
OBTY 4-199  
OLD\_EDFONT 4-201, B-1  
Open streams, list 4-174

Operating system build time 4-16  
Options, standard 3-6  
Orphan objects 4-126  
OS 4-202  
Overstrike 4-202  
Overstrike mode 2-31

## P

Pads  
    close 2-82  
    create 2-13, 2-22  
    move horizontally 2-51  
    move to bottom 2-50  
    move to top 2-54  
    scroll by lines 2-55  
    scroll by pages 2-53  
    set read/write mode 2-58  
    write to a file 2-56  
PAGF 4-203  
Paginate a file 4-203  
Password 4-24  
Paste buffers 2-95, 2-96, 2-98  
    create 2-18  
Paste text from buffer 2-98  
PB 2-50  
PH 2-51  
Pipeline data, save 4-274  
Plain mode protocol 4-258  
PN 2-52  
Points  
    defining 1-1  
Pop a window 2-93  
PP 2-53  
PPON format 4-91, 4-102  
PPRI 4-204  
PRF 4-205  
PRFD 4-212  
Print a file 4-205, 4-212  
Print server 4-216  
Priority of a process 4-204  
Probe network 4-213  
PROBENET 4-213  
Processes  
    continue 2-23  
    create 2-16  
    create background 2-20  
    create remote 4-46  
    create server 2-21  
    display status 4-76, 4-217  
    priority 4-204  
    quit 2-24  
    stop 2-24, 4-253

suspend 2-26  
Program counter histogram  
4-151  
Protected subsystems  
  create 4-51  
  enter 4-114  
  execute manager 4-307  
  set/display attributes  
  4-265  
PRSVR 4-216  
PST 4-217  
PT 2-54  
Push a window 2-93  
PV 2-55  
PW 2-56

## Q

Quit a process 2-24

## R

Range of text 1-4  
Raw characters 2-34  
RBAK 4-219  
RDYM 4-224  
READ 4-225  
Read a backup tape 4-219  
Read a file 2-22  
Read a floppy disk 4-219  
Read a foreign tape 4-231  
Read mode 2-58  
Read user input 4-225,  
  4-227, 4-228  
READC 4-227  
READLN 4-228  
Ready message 4-224  
Receive file from remote host  
  4-255  
Refresh a window 2-60  
Refresh screen 2-59  
Regions 2-25  
  defining 1-2  
Regular expressions 1-2, 1-4  
  set case sensitivity 2-63  
  summary 1-7  
Relative mode 4-278  
Remote transmissions  
  receive 4-255  
  transmit 4-257  
RETURN 4-229

Return from current Shell level  
  4-229  
Return to top of a loop 4-197  
Reverse lines 4-230  
REVL 4-230  
Rights to objects 4-92  
RM 2-57  
RO 2-58  
Routing check 4-230.1  
Routing statistics 4-230.2  
RS 2-59  
RTCHK 4-230.1  
RTSTAT 4-230.2  
RTSVC 4-230.4  
Rubberbanding 2-27, 2-86,  
  2-92  
RW 2-60  
RWMT 4-231

## S

S 2-61  
SALACL 4-238  
SALD 4-239  
SALRGY 4-240  
Salvage  
  ACL structure 4-238  
  directory 4-239  
  disk 4-241, E-1  
  network registry 4-240  
SALVOL 4-241, E-1  
Save data in a pipeline 4-274  
Save transcript pad 2-52  
SC 2-63  
Scrch  
Screen timeout delay 4-242  
Scripts  
  execute (DM) 2-14  
Scroll mode, window 2-94  
SCRTO 4-242  
Search for text  
  abort 2-3  
  blocks (Shell) 4-142  
  in files (DM) 1-2  
  in files (Shell) 4-139,  
  4-142  
SELECT 4-243  
Send alarm messages 4-245  
Send file to remote host 4-257  
SEND\_ALARM 4-245  
SET 4-248  
Set arrow key scale factors  
  2-8  
Set search case sensitivity  
  2-63

Set tabs 2-77  
 Set window color 2-42  
 SETVAR 4-249.1  
 SH 4-250  
 SH8  
 Shell  
   evaluate variables 4-115, 4-116  
   execute 4-250  
   execute DM command 4-304  
   execute script at current level 4-261  
   execution trace 4-305, 4-306  
   handle background output 4-17, 4-18  
   set conditions 4-248  
   verify command lines 4-289, 4-290  
 Shell command format 3-1  
 Shell completion status 4-2  
 Shell conditions 4-248  
 Shell flags  
   -B 4-17, 4-18  
   -E 4-115, 4-116  
   -V 4-289, 4-290  
   -X 4-305, 4-306  
 Show OS version 4-16  
 Shrink a window 2-85, 2-86  
 SHUT 2-64  
   See also EX  
 Shut down system 2-64  
   See also EX  
 SID 4-91  
 Signal a process 4-253  
 SIGP 4-253  
 SIO line, configure 4-270  
 SIORF 4-255  
 SIOTF 4-257  
 SO 2-65  
 Sort files 4-262  
 SOURCE 4-261  
 Special characters (DM) 1-11  
 Special characters (Shell) 3-3  
 Spelling errors, find 4-145  
 SQ 2-66  
 SRF 4-262  
 Standard key names 1-8  
 Status code translation 4-264  
 STCODE 4-264  
 Stop a process 2-24, 4-253  
 Streams, list open 4-174  
 Subject identifier 4-91

SUBS 4-265  
 Substitute text strings 2-61, 2-65, 4-25  
   first occurrence only 2-65  
 Suspend a process 2-26  
 System calendar 4-19, A-1  
 System ready message 4-224

## T

Tab left 2-70  
 Tab right 2-69  
 Tab stop settings 2-77  
 Tabs  
   move left 2-70  
   move right 2-69  
   set 2-77  
 Tapes  
   read backup 4-219  
   read/write foreign 4-231  
   write backup 4-294  
 TB 2-67, 4-269  
 TCTL 4-270  
 TDM 2-68  
 TEE 4-274  
 Test for file existence 4-121  
 Text echoing 2-27  
 TH 2-69  
 THL 2-70  
 TI 2-71  
 Time zone settings 4-281, A-2  
 TL 2-72  
 TLC 4-275  
 TLW 2-73  
 TN 2-74  
 TNI 2-75  
 Touch pad characteristics 4-277  
 TPM 4-277  
 TR 2-76  
 Traceback from a fault 4-269  
 Transliterate characters 4-275  
 Transmit file to remote host 4-257  
 TS 2-77  
 TT 2-78  
 TWB 2-79  
 Types 4-51.1, 4-51.2, 4-72.1, 4-156.1, 4-156.2  
 Types, list installed 4-176  
 TZ 4-281

## U

UCTNODE 4-283  
UCTOB 4-284  
ULKOB 4-285  
Uncatalog a node 4-283  
Uncatalog an object 4-284  
UNDO 2-80  
Undo DM commands 2-80  
Unlock an object 4-285  
User accounts 4-85  
User Change Requests 4-52  
User libraries 4-156

## V

### Variables

assign values from input  
4-225, 4-227, 4-228  
change Shell to Environ-  
ment 4-124  
check existence 4-122  
delete 4-73  
Environment 2-33, 4-124  
evaluate 4-115, 4-116  
list 4-179

VCTL 4-287

VOFF 4-289

### Volumes

dismount 4-74  
mount 4-185

VON 4-290

VSIZE 4-291

VT100 4-292

set terminal characteristics  
4-287  
set window pane size  
4-291  
terminal emulator 4-292

## W

WA 2-81  
WBAK 4-294  
WC 2-82  
WD 4-302  
WDF 2-84  
WG 2-85  
WGE 2-86  
WGRA 2-87

WGRR 2-88  
WH 2-89  
WHILE 4-303  
WHILE loop 4-303  
WI 2-90  
Window boundaries 1-3, 2-84  
define default 2-84  
Window groups  
create/add to 2-87  
delete window from 2-88  
Window modes  
autohold 2-81  
hold 2-89  
scroll 2-94  
visible/invisible 2-90  
Windows  
close 2-82  
copy 2-12  
copy image 2-97  
create 2-13, 2-22  
grow and shrink 2-85,  
2-86  
make icons 2-39  
move 2-91, 2-92  
pop/push 2-93  
refresh single 2-60  
set color 2-42  
WM 2-91  
WME 2-92  
Working directory 4-302  
WP 2-93  
Write backup tape 4-294  
Write a floppy disk 4-294  
Write a foreign tape 4-231  
Write edit pad to file 2-56  
Write files to standard output  
4-20  
Write mode 2-58  
WS 2-94

## X

X/Y coordinates 1-2  
XC 2-95  
XD 2-96  
XDMC 4-304  
XI 2-97  
XOFF 4-305  
XON 4-306  
XP 2-98  
XSUBS 4-307



**READER'S RESPONSE**

We use readers' comments in revising and improving our documents.

Document Title: *DOMAIN System Command Reference*

Order No.: 002547

Revision: 04

Date of Publication: December, 1986

What is the best feature of this manual?

---

---

---

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible.)

---

---

---

---

What type of user are you?

---

How often do you use your system?

---

Nature of your work on the DOMAIN System:

---

Your name

Date

Organization

Street Address

City

State

Zip/Country

No postage necessary if mailed in the U.S. Fold on dotted lines (see reverse), tape, and mail.

cut or fold along dotted line

FOLD



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS      PERMIT NO. 78      CHELMSFORD, MA 01824  
POSTAGE WILL BE PAID BY ADDRESSEE



**APOLLO COMPUTER INC.**  
**Technical Publications**  
**P.O. Box 451**  
**Chelmsford, MA 01824**

FOLD

**READER'S RESPONSE**

We use readers' comments in revising and improving our documents.

Document Title: *DOMAIN System Command Reference*

Order No.: 002547

Revision: 04

Date of Publication: December, 1986

What is the best feature of this manual?

---

---

---

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible.)

---

---

---

---

What type of user are you?

---

How often do you use your system?

---

Nature of your work on the DOMAIN System:

---

Your name

Date

Organization

Street Address

City

State

Zip/Country

No postage necessary if mailed in the U.S. Fold on dotted lines (see reverse), tape, and mail.

cut or fold along dotted line

FOLD



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS      PERMIT NO. 78      CHELMSFORD, MA 01824  
POSTAGE WILL BE PAID BY ADDRESSEE



**APOLLO COMPUTER INC.**  
**Technical Publications**  
**P.O. Box 451**  
**Chelmsford, MA 01824**

FOLD

# Instruction Sheet

**Insert Tabbed Divider Page:**

**Before Page:**

|                            |         |
|----------------------------|---------|
| DM Basics                  | 1-1     |
| DM Task List               | 1-13    |
| DM Command Descriptions    | 2-1     |
| Shell Basics               | 3-1     |
| Shell Task List            | 3-9     |
| Shell Command Descriptions | 4-1     |
| Index                      | Index-1 |

